Thèse n° 10 538

Incorporating Projective Geometry into Deep Learning

Présentée le 12 janvier 2024

Faculté informatique et communications
Laboratoire de vision par ordinateur
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Michał Jan TYSZKIEWICZ

Acceptée sur proposition du jury

École
polytechnique
fédérale
de Lausanne

2024

He who controls the spice, controls the universe.
— Baron Vladimir Harkonnen

To my parents…

Acknowledgements

This is an opportunity to thank those who have shaped me and helped me to get to this priviledged place in life. Due to his most direct impact in that matter, I would like to first thank my supervisor, prof. Pascal Fua, who believed in me back as MSc student in his lab, offered a PhD position and redirected my interests towards 3D vision, a fortunate change which would not have happened otherwise. I am grateful for his trust and the freedom I had to explore emerging topics in this rapidly changing field. I would also like to thank Eduard Trulls, with whom I co-authored two of my papers and even more unpublished projects, and whose technical and goal-oriented mind complemented my free-spirited approach. Finally on that account, I thank Mateusz Koziński, who was a great guide to the lab and perfectly predicted the synergy I would develop with Eduard. On this professional path, I also extend my thanks to Vitto Ferrari, Kevis-Kokitsi Maninis and Stefan Popov who invited me into their team during my 2021 internship (which resulted in RayTran) and to Google as a company, for their research grants which sponsored my work throughout the years.

On a more personal level, I am grateful to have had such a wonderful team of friends in Lausanne — Lars, Nicolas, Plouton, Aditya and others, all lead by the tireless Dina. This was vital to my sanity, especially during the pandemic, and I congratulate the team of EDIC for their stellar efforts to integrate us as friends, and not just coworkers. Similarly, I am extremely grateful to Ewa and Stanley Hesse, my landlords who became more of a second set of Swiss parents.

Speaking of parents, Basia and Jurek, I am obviously indebted to them, my sister Olga and my family who shaped me as I am, as well as to a long list of exceptional teachers I am lucky to have encountered along the way. This starts with Dr Dominika Depta-Marel who taught me how to properly formulate questions, Anna Olszańska who made me like mathematics, the incredible team at Bednarska, comprising of Dorota, Jędrek, Kasia, Grześ, Gabi and others who showed me the diversity of life — both biological and everyday — and the team of prof. Wojciech Grochala at University of Warsaw, in particular dr Tomasz Jaroń and dr Piotr Leszczyński for introducing me to scientific research.

Finally I would like to thank my friends back from Poland, who despite my lengthy stay abroad are always there to listen: Ola, Ignacy, Marcin, Maciek, Hela and many others.

Lausanne, December 22, 2023 Michał Tyszkiewicz

Abstract

In this thesis we explore the applications of projective geometry, a mathematical theory of the relation between 3D scenes and their 2D images, in modern learning-based computer vision systems. This is an interesting research question which contradicts the recent trend to forgo such domain knowledge in favor of learning everything directly from data. We show how to use these robust mathematics where applicable while maximally leveraging data for the remaining aspects.

The thesis extends three peer-reviewed papers. In the first, we introduce an algorithm to extract local image features, a technique of matching related regions across images. Unlike in standard supervised learning, we do not define the features through examples but rather their desired properties. We leave it to the training procedure to find a conforming algorithm. This shows an application of projective geometry for *supervision* of neural networks. We then turn to two cases of using projective geometry in the network architecture. In one, we present a method to deduce indoor scene layouts from video walkthroughs. We constrain the Transformer, a computationally intensive task-agnostic learning system, by using relevant geometry to significantly reduce its processing time and enhance memory efficiency. In the last paper, we address the challenge of reversing the 3D-to-2D projection in a generative setting. By offering multiple potential 3D reconstructions based on a 2D view, we acknowledge the inherent uncertainties of this inversion. Each chapter provides a thorough review of existing literature and outlines potential avenues for future research in the domain.

Keywords: computer vision, 3D vision, projective geometry, transformers, diffusion models, reinforcement learning, local features, object detection, point clouds

Zusammenfassung

In dieser Arbeit untersuchen wir die Anwendungen der projektiven Geometrie, einer mathematischen Theorie der Beziehung zwischen 3D-Szenen und ihren 2D-Bildern, in modernen lernbasierten Computer-Vision-Systemen. Dies ist eine interessante Forschungsfrage, die dem aktuellen Trend widerspricht, auf solches Domänenwissen zu verzichten und stattdessen alles direkt aus Daten zu lernen. Wir zeigen, wie man diese robuste Mathematik anwenden kann und gleichzeitig die Daten für die verbleibenden Aspekte maximal nutzen.

Die Arbeit erweitert drei peer-reviewte Arbeiten. Im ersten Schritt stellen wir einen Algorithmus zum Extrahieren lokaler Bildmerkmale vor, eine Technik zum Abgleichen verwandter Regionen in Bildern. Anders als beim standardmäßigen überwachten Lernen definieren wir die Funktionen nicht anhand von Beispielen, sondern anhand ihrer gewünschten Eigenschaften. Wir überlassen es dem Trainingsverfahren, einen konformen Algorithmus zu finden. Dies zeigt eine Anwendung der projektiven Geometrie zur Überwachung neuronaler Netze. Anschließend wenden wir uns zwei Fällen der Verwendung projektiver Geometrie in der Netzwerkarchitektur zu. In einem davon stellen wir eine Methode vor, um Innenszenenlayouts aus Videobesichtigungen abzuleiten. Wir schränken den Transformer, ein rechenintensives aufgabenunabhängiges Lernsystem, durch die Verwendung relevanter Geometrie ein, um seine Verarbeitungszeit erheblich zu verkürzen und die Speichereffizienz zu verbessern. Im letzten Artikel befassen wir uns mit der Herausforderung, die 3D-zu-2D-Projektion in einer generativen Umgebung umzukehren. Indem wir mehrere mögliche 3D-Rekonstruktionen basierend auf einer 2D-Ansicht anbieten, erkennen wir die inhärenten Unsicherheiten dieser Inversion an. Jedes Kapitel bietet einen gründlichen Überblick über die vorhandene Literatur und skizziert mögliche Wege für zukünftige Forschung auf diesem Gebiet.

Contents

1 Introduction

The space we inhabit is intrinsically three-dimensional and our lives revolve around 3D environments — from the way we perceive objects to our interaction with the surroundings. Until recently, the most influencial works in visual computation and analysis has been exploring the two dimensions with convolutional neural networks. The famous works of the time, including ResNet [75], MaskRCNN [77], YOLO [180] or U-Net [189], have been evaluating on 2D benchmarks such as ImageNet [45] and COCO [126]. While these representations of our environment have allowed for tremendous progress, in the end they are a reductionist perspective of the reality. As the techniques for working with 2D became more established, a new wave of interest in 3D vision has emerged (a notable exception and driver of 3D vision research has always been human pose estimation [18, 88]), with the goal of understanding the world around us in its true form. Important recent concepts in this space include implicit representations, such as neural radiance fields [147, 153, 100] and signed distance functions [162, 144, 34], vision backbones which do not rely on image raster layout [51] and large-scale 3D datasets [181, 243]. This thesis is a part of that community-wide endeavor, exploring the uses of *projective geometry*, a mathematical theory relating 3D scenes to their 2D views, in inferring spatial information from these views.

Today, cameras are one of the most pervasive and formidable sensors in our arsenal. They are affordable and ubiquitous, dotting our smartphones, streets, and spaces. The surge in camera technology and availability has enabled a broad spectrum of applications, from social photography to scientific analysis. Contrastingly, other sensors capable of capturing 3D information, like haptic devices, radar, or long range lidar, are neither as widespread nor as cost-effective as cameras. Yet, there is an inherent challenge. The process of image formation is fraught with loss. From a multidimensional world, we get a 2D snapshot. Therefore, any endeavor to extrapolate 3D information from images demands we find a solution which is coherent across multiple images, leverage prior knowledge about common shapes and sizes, or both. This setting of incorporating prior beliefs and multiple lossy observations to infer their causes is common across sciences and known as an *inverse problem.* Our reliance on cross-referencing the images and making assumptions about the scene remains true whether our end-goal is the inverse solution itself, for example in photogrammetry [202] and robotic simultaneous localization and mapping

(SLAM) [61], or we wish to solve a derived task, like 3D object detection or object placement in extended reality applications.

1.1 Presentation of the works

The key to solving inverse problems is understanding the forward process, that is how the causes influence the observations. In the context of 3D vision, the causes are the geometric layout of the imaged scene, the lighting, and the appearance properties of the materials involved. The forward process itself is known as *image formation* and involves the physics of light bouncing around the scene, culminating in its registration onto the camera's sensor. This last stage is governed by the mathematical theory of *projective geometry* and in this thesis, I work on leveraging it as a fundamental constraint for recovering geometries from images. I showcase three distinct yet interconnected approaches to harnessing this mathematical model, which are briefly outlined below and expanded upon in the subsequent chapters.

1.1.1 DISK [221]

Local features are a class of algorithms which divide an image into a set of small components and their use is for *matching* related parts of images [151, 74, 133]. Unlike superpixels [1], which aim at exhaustive decomposition into internally homogeneous regions, local features are sparse and focus on *salient* regions which ideally correspond to 3D *landmarks*. This means that across imaging angles and conditions we expect to detect a similar set of landmarks, a property known as *repeatability*. Thanks to their saliency, local features can be described with a small appearance-based signature (called *descriptor*) which is similar across the views and dissimilar from different landmarks. Put together, a local feature algorithm extracts from an image a set of *keypoints* which can then be matched with those of a different image. This is a fundamental task which enables estimation of the relative viewing angles between two (stereo) or more (Structure from Motion, SfM) images and the 3D location of the landmarks corresponding to individual features.

Spatial relaxations. At the time of writing DISK, in 2019, the wave of deep learning had swept most of computer vision, with keypoint detection for local features being one of the few remaining bastions of handcrafted algorithms. The reason is twofold. Firstly, good local features are defined through their properties (reliability of detection, saliency of appearance) rather than by example, making them unsuitable for supervised learning. Manual annotation of images with thousands of keypoints each is impractical and doing so automatically means the learned algorithm will merely imitate that automatic labeler, providing no benefit. Secondly, there is a fundamental incompatibility between neural networks, continous and differentiable functions optimized through gradient descent, and the discrete task of picking a subset of image pixels as the keypoint locations. A continous relaxation is necessary for optimization and most prior work either trained detection solely for repeatability [46], possibly resulting in non-salient detections, or focused on relaxing the location of keypoints [161, 52, 241, 183], which is at odds with the requirements of high keypoint precision and caused problems with computing the descriptors at those ill-defined locations. On the other hand, given well defined keypoints, learning descriptors is a well studied task of *metric learning*. Overall, the state of art prior to DISK was that handcrafted keypoint detectors paired with learned descriptors still ruled supreme [95].

Probabilistic relaxation. DISK took a different approach than prior work and relaxed the *probability* of selecting a pixel as a keypoint, rather than its location. While training, we sample a set of keypoints, making them discrete objects with precise location, allowing for computing their descriptors. We extend this probabilistic formulation to also cover matching features across two images, yielding a unified model. We can assign a reward R for each feature match and then use tools from reinforcement learning [229] to estimate of the gradient of its expectation w.r.t. network parameters θ, $\nabla_\theta \mathbb{E}[R]$. This removes the requirements of differentiability from the reward formulation and allows for more direct search for the optimal solution to feature detection.

Projective geometry. Given the freedom to design a reward function $R(k_a, k_b)$ which scores the desirability of a match between the two keypoints in two images, we turn to projective geometry for a principled formulation. Given a dataset of large collections of images alongside their estimated poses and depth maps, we unproject k_a from 2D to 3D and then project it to the other image, obtaining k'_a. We can measure the distance $|k_b - k'_a|$ and symmetrically $|k_a - k'_b|$ to check if the geometry of the scene allows for the two detections to coincide with the same 3D landmark. The reward R is then defined as positive if they do and negative otherwise. This simple formulation imposes no properties other than those in the definition of a local feature, allowing the optimization process to find the best detection and description criteria. This is in contrast to prior work which forced keypoints to be corners [46] or semantic feature extrema [52]. This freedom results in discovery of image features with rich descriptors (Fig. 1.1) substantially superior to those of prior work. We also use an alternative reward definition based on epipolar geometry to sidestep the need for accurate depth maps and find that this too results in performant local features.

1.1.2 RayTran [223]

3D object detection from multiple views. The well-studied task of detecting objects in images [126, 182, 25] is defined as mapping an image to a set of objects with a bounding box and a set of metadata, commonly an object class and a segmentation mask which provides a more precise outline. Despite the recent research on open-world object detection [96], most works focus on detecting a closed set of categories of interest. A natural extension is to detect objects in 3D, extending the notion of a bounding box to a rectangular cuboid with defined 3D position, rotation and scale, bringing a total of nine degrees of freedom (*9DOF*). As already mentioned, a

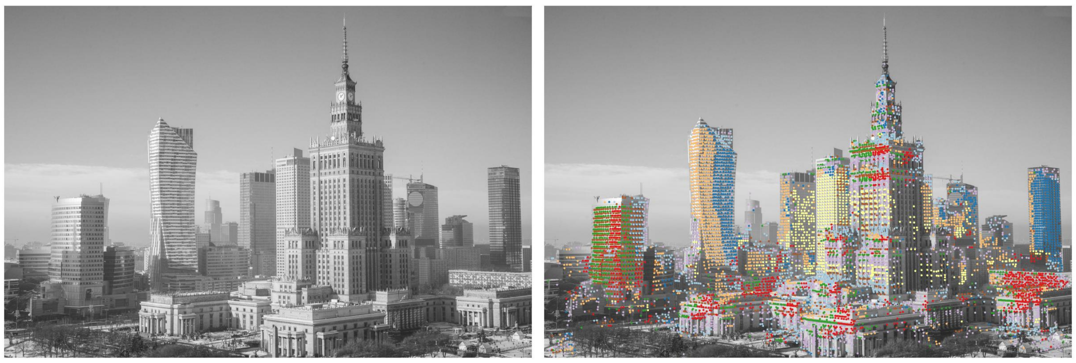
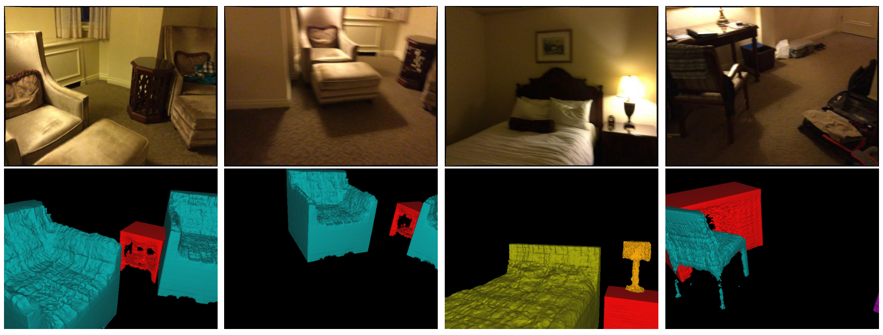
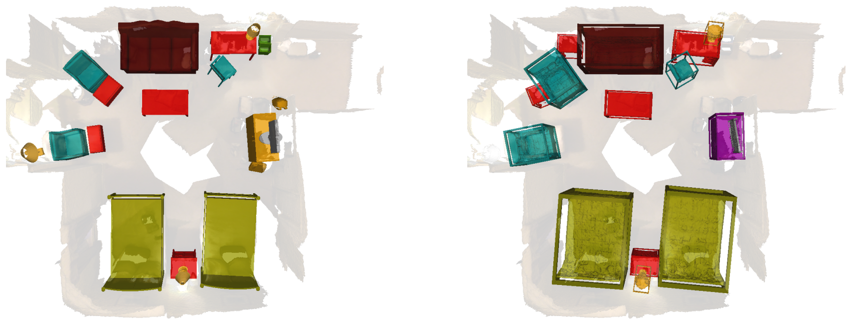


Figure 1.1: DISK features. Left: input image, right: 8196 local features split into 8 clusters by applying K-means on their descriptors and color-coding accordingly. This visualization reveals the cues used by the algorithm, for example the distinction of the left and right side of facades (green/red). Image licensed from iStockPhoto.



(a) **Top**: selection of input images, **bottom**: predictions rendered from camera's POV.



(b) **Left**: GT scene layout, **right**: RayTran predictions. Both are overlaid on scene scan (not used by RayTran); object colors show semantic classes.

Figure 1.2: RayTran first builds an internal 3D representation of the scene from input views (**top**) and treats it as an input "image" to an object detector to find objects of interest (**bottom**).

single image does not contain enough information to disambiguate all these parameters, requiring additional sources of information. Common assumptions are availability of depth maps, acquired with extra sensors, or outright 3D scans of the scene of interest. In RayTran we make the assumption of availability of multiple posed 2D views of the scene. Given a video, for many scenes this can be solved with SfM or SLAM, and in case of difficulty the scene can be augmented with markers to help with tracking the camera position [63].

Detection aggregation via object tracking. Prior work in the field focused on object tracking approaches. This means that instead of holistically modelling the scene, objects of interest are detected in each image independently, with all the imprecisions and ambiguities, and the detections are later associated across views. This process aims at connecting all detections of the same 3D object, allowing for resolving pose and appearance ambiguities by for example averaging the individual pose estimates. The issue with this approach is that when multiple objects are visually similar, as is the case with indoors ever since IKEA, the algorithm encounters a chicken and egg problem. When appearance cues are weak, the association algorithm must distinguish objects by pose (in particular location) which is unreliable until after the associations are made. This gets more complicated as objects come in and out of the video's field of view, precluding use of smooth appearance changes as a cue. Additionally, it is difficult to connect the detector with the tracker in a differentiable way, to enable joint learning and co-adaptation of the two stages. We identify these factors as limiting the performance of the prior work.

Scene representation. RayTran takes a very different approach to the problem, starting with 3D fusion before any detections are made. We perform no detections on the individual frames and instead centralize the information in a geometrically sound representation of the scene. Specifically, we initialize a 3D voxel grid to represent the scene content at any (x, y, z) location and populate it with information derived in a learned fashion from individual views. This is the inverse problem of inferring a full scene from projections. We put no explicit requirements on the kind of features stored, leaving it to the network to aggregate class semantics, appearances, monocular pose and depth estimates, and other cues useful in 3D object detection. Once ready, the scene representation is treated similarly as an image would in standard 2D object detection, yielding 3D object detections in a single pass.

Geometric constraints. The key to making RayTran work is again in projective geometry. Unlike DISK, we use it to inform the network's internal operation rather than supervise training. Specifically, we do a simple modification at the stage of building the scene representation. We restrict the information flow between the input view pixels and scene representation voxels in such a way that a voxel is only directly connected to pixels which coincide with its projection, or correspondigly that pixels can only inform voxels which lie along their viewing ray. This single restriction serves two purposes. Firstly, it provides the only source of geometry input to the network by blocking exchange of information among irrelevant paths. This leaves the remaining ambiguity of depth (along the viewing ray) for the network to resolve by intersecting multiple viewing frusta. Secondly, by focusing only on useful paths it reduces the computational and memory requirements of the method by orders of magnitude, making the approach feasible.

1.1.3 GECCO [222]

Point cloud generation. Point clouds are one of the simplest ways to represent shapes, used especially for LiDAR scans and SfM reconstructions [202]. They are fundametally *sets*, with no defined ordering of points. Additionally, in the case of scanning, the specific point locations are also a nuisance factor, arising from uniform sampling of the underlying surfaces. This makes point clouds tricky from the algorithmic perspective, and especially for generation, requiring models which can reason both the high-level shape as well as about individual points and their permutation. It is nevertheless an interesting problem, due to the scale of available point cloud datasets [125] and their emerging applications [100]. In our work we propose a novel formulation for generating point clouds and show its ability to scale to large real world datasets [243]. We also show how to condition the generation on images, resulting in high quality 3D hypotheses about scene geometry, which reflect the uncertainty of this inverse problem through the diversity of generated samples, as shown in Fig. 1.3.

Generative modelling. Generative modelling is a task in machine learning where we are given a set of discrete samples from a target distribution $P(y)$ and asked to approximate that distribution $\hat{P} \approx P$ to allow for generating novel samples $\hat{y} \sim \hat{P}$. This is in contrast to most supervised tasks where it suffices to match some statistical properties of P like the mean or mode. Supervised learning often makes simplifying assumptions like Gaussian distribution of errors in the case of least squares regression or independence of individual variables in the case of pixel-wise classification. These models exhibit similar *statistics* to the true P but possibly assign very different probabilities to individual events, making them unsuitable for sampling. An example are image segmentation models, trained to estimate the class probabilities of each pixel individually. Due to them not modelling inter-pixel correlations, they may produce discontinuous predictions, even when trained exclusively with contiguous masks. For many tasks, such as estimating the area of an object, this is an acceptable price for speed and simplicity, but in generative modelling the goal is to approximate P as closely as possible, not just its statistics.

Factorized models of shape and points. Starting with the seminal PointFlow [236], the dominant line of work on point cloud modelling explicitly recognized the distinction between the underlying shape and the variation specific to a given point cloud. Their models factorized the generative process in two steps, first sampling an unstructured latent shape code s, and then sampling individual points p_i conditionally on s and independently of each other. This leads to two problems. Firstly, unless s describes that surface very precisely (which is difficult with fairly small and unstructured latent codes), $P(p_i|s)$ may encode residual uncertainty about the object

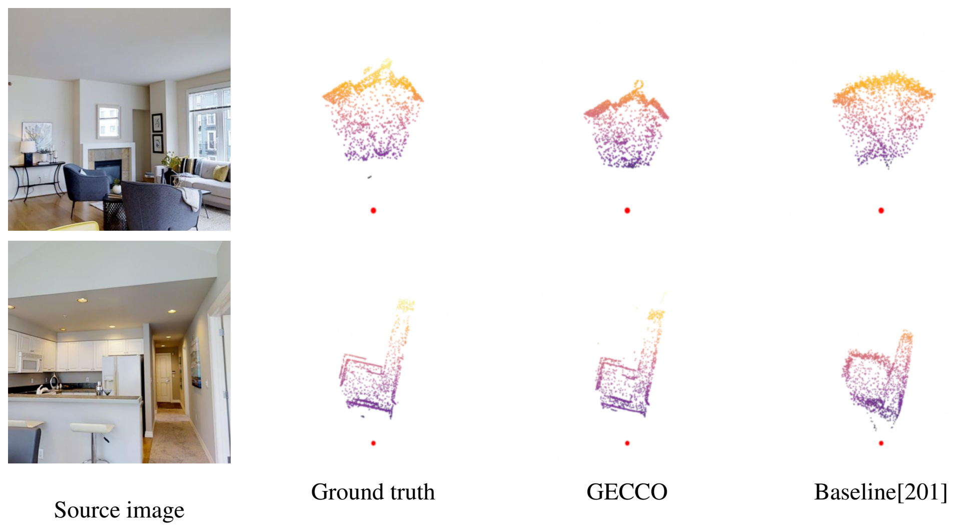


Figure 1.3: Input images and associated point clouds, rendered from bird's eye view. Red dot marks the original camera. Despite uncertainty and error (mis-estimated maximum depth) GECCO, thanks to its generative nature, creates a plausible scene layout — preserving right angles, flat surfaces and proportions. A monocular depth baseline, trained with an L_2 objective, ignores the relative positions of points, resulting in curved walls and implausible object shapes.

surface itself, and not just about an individual point's location on it. This would result in light "fuzziness" of points sampled independently from each other. Secondly, the probability of a point cloud in this formulation is intractable to compute due to requiring an integral over all possible s. This makes it impossible to train the model by maximum likelihood — the usual training strategy for their class of models, the normalizing flows [184, 50]. For this reason PointFlow and follow-up papers [135, 244] set up involved autoencoding schemes to obtain the shape code s for each training example, at training time. This makes the entire system more complicated and brittle at training, requiring the practitioner to tune the relative importance of the components in their loss function.

Joint probability density. Our first contribution lies in using the recent continous diffusion models [211, 98] to model the point cloud jointly. Continuous diffusion models synthesize new samples by solving a differential equation involving a learned neural operator $f_\theta(x,t)$. Although related to normalizing flows used by PointFlow, it does not require solving the equation *at training time*, making the framework much more scalable. It lets us model the entire point cloud generation process jointly, including point-point inter-dependencies. This in turn lets us rid of autoencoding and make the training procedure simpler, the sampling faster, and gives the ability to estimate point cloud probabilities. At the same time, thanks to properties of the neural network we choose for parametrization of f_θ, we retain the ability to efficiently sample dense point clouds by treating the point cloud itself as its shape latent s and conditionally sample additional points.

Geometric conditioning. Our second contribution, in line with the topic of this thesis, leverages projective geometry. Our specific goal is to learn image-conditioned densities over point clouds to facilitate reasoning about the uncertainty of monocular shape estimates and occluded regions. To this end we design a continous diffusion model which extends $f_\theta(x, t)$ with view information v to $f_\theta(x, t, v)$, continuously informing the equation about the relation of the current shape x_t to the image content. Specifically, we start by feeding the image to a CNN to extract dense features. Then, at each step of solving the sampling equation, we use camera parameters to project each point p_i to the image plane and look up previously obtained features at the corresponding location. This helps the network precisely align the point cloud with the image but provides no information about depth, lost during image formation. That aspect is handled by the network learning the sizes of common objects from data and using those priors along with apparent sizes in the image to estimate their depth. Since this process happens in 3D and over the entire point cloud jointly, the network is able to formulate a coherent hypothesis about the sizes, depths and layout of all visible objects. Since the model is generative, the samples are *plausible* — for example in respecting 90° wall angles. This is compared to a regression-based baseline shown in Fig. 1.3 which tries to estimate the depth in each part of the room individually, resulting in “bent” walls. Overall, GECCO is a conditional generative model for point clouds characterized by exceptional geometric fidelity. It expresses the uncertainty of the monocular 2D-to-3D inverse problem, such as the scale-depth relation, through the diversity of generated samples.

1.2 Organization of the thesis

The following chapters 2, 3 and 4 expand on the three algorithms outlined above. Aside from the content of these original works, each chapter is followed by an extended background and discussion of the motivation behind the project. We also discuss the reception and impact of the works as well as lessons learned, in particular negative results, and propose future work directions in the respective topics. Finally, in chapter 5, we summarize the contributions of this thesis.

2 DISK

"DISK: Learning local features with policy gradient" by Michał Tyszkiewicz, Pascal Fua, Eduard Trulls in NeurIPS2020. Candidate's contribution: system design, implementation.

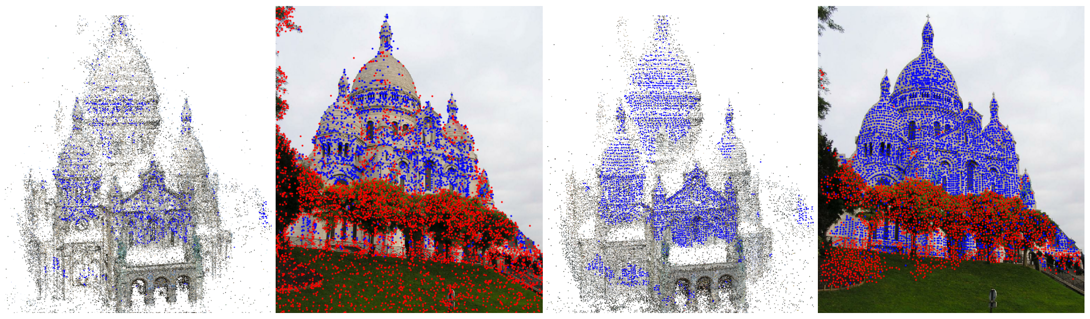
2.1 Abstract

Local feature frameworks are difficult to learn in an end-to-end fashion, due to the *discreteness* inherent to the selection and matching of sparse keypoints. We introduce DISK (DIScrete Keypoints), a novel method that overcomes these obstacles by leveraging principles from Reinforcement Learning (RL), optimizing end-to-end for a high number of correct feature matches. Our simple yet expressive probabilistic model lets us keep the training and inference regimes close, while maintaining good enough convergence properties to reliably train from scratch. Our features can be extracted very densely while remaining discriminative, challenging commonly held assumptions about what constitutes a good keypoint, as showcased in Fig. 2.1, and deliver state-of-the-art results on three public benchmarks.

2.2 Introduction

Local features have been a key computer vision technology since the introduction of SIFT [133], enabling applications such as Structure-from-Motion (SfM) [3, 79, 204], SLAM [154], re-localization [138], and many others. While not immune to the deep learning "revolution", 3D reconstruction is one of the last bastions where sparse, hand-crafted solutions remain competitive with or outperform their dense, learned counterparts [203, 199, 95]. This is due to the difficulty of designing end-to-end methods with a differentiable training objective that corresponds well enough with the downstream task.

While patch descriptors can be easily learned on predefined keypoints [208, 217, 149, 218, 54], joint detection and matching is harder to relax in a differentiable manner, due to its computational complexity. Given two images A and B with feature sets F_A and F_B, matching them is $O(|F_A|\cdot|F_B|)$. As each image pixel may become a feature, the problem quickly becomes intractable. Moreover, the "quality" of a given feature depends on the rest, because a feature that is very similar to others is less distinctive, and therefore less useful. This is hard to account for during training.



(a) Upright Root-SIFT [133, 5]
179k landmarks, 22.3 observations/landmark

(b) DISK (ours)
190k landmarks, 30.0 observations/landmark

Figure 2.1: **SIFT vs. DISK in SfM.** We reconstruct "Sacre Coeur" from 1179 images [95] with COLMAP. For Upright Root-SIFT (left) and DISK (right) we show a point cloud and one image with its keypoints. Landmarks, and their respective keypoints, are drawn in **blue**. Keypoints which do not create landmarks are drawn in **red**. Our features can be extracted (and create associations) on seemingly textureless regions where SIFT fails to, producing more landmarks with more observations.

We address this issue by bridging the gap between training and inference to fully leverage the expressive power of CNNs. Our backbone is a network that takes images as input and outputs keypoint 'heatmaps' and dense descriptors. Discrete keypoints are sampled from the heatmap, and the descriptors at those locations are used to build a distribution over feature matches across images. We then use geometric ground truth to assign positive or negative rewards to each match, and perform gradient descent to maximize the expected reward $\mathbb{E}\sum_{(i,j)\in M_{A\leftrightarrow B}} r(i \leftrightarrow j)$, where $M_{A\leftrightarrow B}$ is the set of matches and r is per-match reward. In effect, this is a policy gradient method [229].

Probabilistic relaxation is powerful for discrete tasks, but its applicability is limited by the fact that the expected reward and its gradients usually cannot be computed exactly. Therefore, noisy Monte Carlo approximations have to be used instead, which harms convergence. We overcome this difficulty by careful modeling that yields analytical expressions for the gradients. As a result, we can benefit from the expressiveness of policy gradient, narrowing the gap between training and inference and ultimately outperforming state-of-the-art methods, while still being able to train models from scratch.

Our contribution therefore is a novel, end-to-end-trainable approach to learning local features that relies on policy gradient. It yields considerably more accurate matches than earlier methods, and this results in better performance on downstream tasks, as illustrated in Fig. 2.1 and Sec. 3.5.

2.3 Related Work

The process of extracting local features usually involves three steps: finding a keypoint, estimating its orientation, and computing a description vector. In traditional methods such as SIFT [133] or

SURF [12], this involves many hand-crafted heuristics. The first wave of local features involving deep networks featured descriptors learned from patches extracted on SIFT keypoints [242, 73, 208] and some of their successors, such as HardNet [149], SOSNet [218], and LogPolarDesc [54], are still state-of-the-art. Other learning-based methods focus on keypoints [226, 200, 113] or orientations [238], or merge the two notions entirely [38].

These methods attack a single element of this process. Others have developed end-to-end-trainable pipelines [240, 46, 161, 52, 183] that can optimize the whole process and, hopefully, improve performance. However, they either use inexact approximations to the true objective [46, 183], break differentiability [161] or make big assumptions, such as extrema in descriptor space making good features [52].

Three recent approaches are attempting to bridge the gap between training and inference in a spirit close to ours. GLAMpoints [219] seeks to estimate homographies between retinal images and use Reinforcement Learning (RL) methods to find keypoints that are correctly matched by SIFT descriptors. Since matching is deterministic, Q-learning can be used to regress for the expected reward of each keypoint, rather than optimize directly in policy space. Using hand-crafted descriptors and only addressing the detection problem was motivated by domain-specific requirements of strong rotation equivariance, which most learned models lack. While it makes sense in the specific scenario it was developed for, it limits what the method can do. Similarly, [39] also uses handcrafted descriptors and learns to predict the probability that each pixel would be successfully matched with those. Their approach therefore inherits many of the limitations of GLAMpoints.

Reinforced Feature Points [17] address the more difficult issue of learning with a general non-differentiable objective for the purpose of camera pose estimation, with RANSAC in the loop. Unfortunately, supervising all detection and matching decisions with a single reward means that this approach suffers from weak training signal, an endemic RL problem, and has to rely on pre-trained models from [46] that can only be fine-tuned. Our method can be seen as a relaxation of their approach, where we train for a surrogate objective: finding many correct feature matches. This allows for substantially more robust training from scratch and yields better downstream results.

2.4 Method

Given images A and B, our goal is first to extract a set of local features F_A and F_B from each and then match them to produce a set of correspondences $M_{A\leftrightarrow B}$. To learn how to do this through reinforcement learning, we redefine these two steps probabilistically. Let $P(F_I|I,\theta_F)$ be a distribution over sets of features F_I, conditional on image I and feature detection parameters θ_F, and $P(M_{A\leftrightarrow B}|F_A,F_B,\theta_M)$ be a distribution over matches between features in images A and B, conditional on features F_A, F_B, and matching parameters θ_M. Calculating $P(M_{A\leftrightarrow B}|A,B,\theta)$ and its derivatives requires integrating the product of these two probabilities over all possible

F_A, F_B, which is clearly intractable. However, we can estimate gradients of expected reward $\nabla_\theta \mathbb{E}_{M_{A\leftrightarrow B}\sim P(M_{A\leftrightarrow B}|A,B,\theta)} R(M_{A\leftrightarrow B})$ via Monte Carlo sampling and use gradient ascent to maximize that quantity.

Feature distribution $P(F_I|I,\theta_F)$. Our feature extraction network is based on a U-Net [188], with one output channel for detection and N for description. We denote these feature maps as $\mathbf{K}$ and $\mathbf{D}$, respectively, from which we extract features $F=\{K,D\}$. We pick N=128, for a direct comparison with SIFT and nearly all modern descriptors [133, 149, 136, 218, 54, 183].

The detection map $\mathbf{K}$ is subdivided into a grid with cell size $h\times h$, and we select at most one feature per grid cell, similarly to SuperPoint [46]. To do so, we crop the feature map corresponding to cell u, denoted $\mathbf{K}^u$, and use a softmax operator to normalize it. Our probabilistic framework samples a pixel p in cell u with probability $P_s(p|\mathbf{K}^u)=\text{softmax}(\mathbf{K}^u)_p$. This detection proposal p may still be rejected: we accept it with probability $P_a(\text{accept}_p|\mathbf{K}^u)=\sigma(\mathbf{K}^u_p)$, where $\mathbf{K}^u_p$ is the (scalar) value of the detection map $\mathbf{K}$ at location p in cell u, and σ is a sigmoid. Note that $P_s(p|\mathbf{K}^u)$ models *relative* preference across a set of different locations, whereas $P_a(\text{accept}_p|\mathbf{K}^u)$ models the *absolute* quality for location p. The total probability of sampling a feature at pixel p is thus $P(p|\mathbf{K}^u)=\text{softmax}(\mathbf{K}^u)_p\cdot\sigma(\mathbf{K}^u_p)$. Once feature locations $\{p_1,p_2,...\}$ are known, we associate them with the l_2-normalized descriptors at this location, yielding a set of features $F_I=\{(p_1,\mathbf{D}(p_1)),(p_2,\mathbf{D}(p_2)),...\}$. At inference time we replace softmax with $\arg\max$, and σ with the sign function. This is again similar to [46], except that we retain the spatial structure and interpret cell $\mathbf{K}^u$ in both a relative and an absolute manner, instead of creating an extra *reject* bin.

Match distribution $P(M_{A\leftrightarrow B}|F_A,F_B,\theta_M)$. Once feature sets F_A and F_B are known, we compute the l_2 distance between their descriptors to obtain a distance matrix $\mathbf{d}$, from which we can generate matches. In order to learn good local features it is crucial to refrain from matching ambiguous points due to repeated patterns in the image. Two solutions to this problem are cycle-consistent matching and the ratio test. Cycle-consistent matching enforces that two features be nearest neighbours of each other in descriptor space, cutting down on the number of putative matches while increasing the ratio of correct ones. The ratio test, introduced by SIFT [133], rejects a match if the ratio of the distances between its first and second nearest neighbours is above a threshold, in order to only return confident matches. These two approaches are often used in conjunction and have been shown to drastically improve results in matching pipelines [13, 95], but they are not easily differentiable.

Our solution is to relax cycle-consistent matching. Conceptually, we draw *forward* (A→B) matches for features $F_{A,i}$ from categorical distributions defined by the rows of distance matrix $\mathbf{d}$, and *reverse* (A←B) matches for features $F_{B,j}$ from distributions based on its columns. We declare $F_{A,i}$ to match $F_{B,j}$ if both the forward and reverse matches are sampled, *i.e.*, if the samples are consistent. The forward distribution of matches is given by $P_{A\rightarrow B}(j|\mathbf{d},i)=\text{softmax}\left(-\theta_M\mathbf{d}(i,\cdot)\right)_j$, where θ_M is the single parameter, the inverse of the softmax temperature. $P_{A\leftarrow B}$ is analogously defined by $\mathbf{d}^T$.

It should be noted that, given features F_A and F_B, the probability of any particular match can be computed *exactly*: $P(i \leftrightarrow j) = P_{A\to B}(i|\mathbf{d}, j) \cdot P_{A\leftarrow B}(j|\mathbf{d}, i)$. Therefore, as long as reward R factorizes over matches as $R(M_{A\leftrightarrow B}) = \sum_{(i,j)\in M_{A\leftrightarrow B}} r(i \leftrightarrow j)$, given F_A and F_B, we can compute *exact* gradients $\nabla_{D,\theta_M} \mathbb{E} R(M_{A\leftrightarrow B})$, without resorting to sampling. This means that the matching step does not contribute to the overall variance of gradient estimation, unlike in [17], which we believe to be key to the good convergence properties of our model. Finally, one can also replace our matching relaxation with a non-probabilistic loss like in [149]. While it may be superior for descriptors alone, our solution upholds the probabilistic interpretation of the pipeline, making the hyperparameters $(\lambda_{tp}, \lambda_{fp}, \lambda_{kp})$ easy to tune and naturally integrating with the gradient estimation in keypoint detection.

Reward function $R(M_{A\leftrightarrow B})$**.** As stated above, if the reward $R(M_{A\leftrightarrow B})$ can be factorized as a sum over individual matches, the formulation of $P(M_{A\leftrightarrow B}|F_A, F_B, \theta_M)$ allows for the use of closed-form formulas while training. For this reason we use a very simple reward, which rewards correct matches with λ_{tp} points and penalizes incorrect matches with λ_{fp} points. Let's assume we have ground-truth poses and pixel-to-pixel correspondences in the form of depth maps. We declare a match *correct* if depth is available at both $p_{A,i}$ and $p_{B,j}$, and both points lie within ϵ pixels of their respective reprojections. We declare a match *plausible* if depth is not available at either location, but the epipolar distance between the points is less than ϵ pixels, in which case we neither reward nor penalize it. We declare a match *incorrect* in all other cases.

Gradient estimator. With R factorized over matches and $P(i \leftrightarrow j|F_A, F_B, \theta_M)$ given as a closed formula, the application of the basic policy gradient [229] is fairly simple: with F_A, F_B sampled from their respective distributions $P(F_A|A, \theta_F), P(F_B|B, \theta_F)$ we have

$$\nabla_\theta \mathop{\mathbb{E}}_{M_{A\leftrightarrow B}} R(M_{A\leftrightarrow B}) = \mathop{\mathbb{E}}_{F_A, F_B} \sum_{i,j} \left[P(i \leftrightarrow j|F_A, F_B, \theta_M) \cdot r(i \leftrightarrow j) \cdot \nabla_\theta \Gamma_{ij} \right], \tag{2.1}$$

$$\text{where } \Gamma_{ij} = \log P(i \leftrightarrow j|F_A, F_B, \theta_M) + \log P(F_{A,i}|A, \theta_F) + \log P(F_{B,j}|B, \theta_F).$$

The summation above is non-exhaustive, missing the case of i not being matched with any j: since we award non-matches 0 reward, they can be safely ommited from the gradient estimator. Having a closed formula for $P(i \leftrightarrow j|F_A, F_B, \theta_M)$ along with R being a sum over individual matches allows us to compute the sum in equation 2.1 exactly, which in the general case of REINFORCE [229] would have to be replaced with an empirical expectation over sampled matches, introducing variance in the gradient estimates. In our formulation, the only sources of gradient variance are due to mini-batch effects and approximating the expectation w.r.t. choices of F_A, F_B with an empirical sum.

It should also be noted that our formulation does not provide the feature extraction network with any supervision other than through the quality of matches those features participate in, which

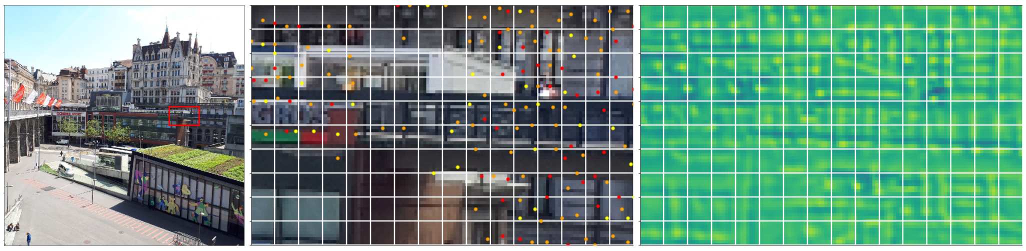


Figure 2.2: **Non-Maxima Suppression vs Grid-based sampling.** We demonstrate the benefits of replacing the 1-per-cell sampling approach used during training with simple NMS at inference time. For a small region of an image (left), marked by the red box, we show the features chosen through NMS (middle) and the 'heatmap' **K** (right), overlaid by the grid. Notice how maxima can be cut by cell boundaries. Keypoints are sorted by "score" and color-coded: the top third are drawn in **red**, the next third in **orange**, and the rest in yellow. Each cell contains at most two very salient (red) features.

means that a keypoint which is never matched is considered neutral in terms of its value. This is a very useful property because keypoints may not be co-visible across two images, and should not be penalized for it as long as they do not create incorrect associations. On the other hand, this may lead to many unmatchable features on clouds and similar non-salient structures, which are unlikely to contribute to the downstream task but increase the complexity in feature matching. We address this by imposing an additional, small penalty on each sampled keypoint λ_{kp}, which can be thought of as a regularizer.

Inference. Once the models have been trained we discard our probabilistic matching framework in favor of a standard cycle-consistency check, and apply the ratio test with a threshold found empirically on a validation set. Another consideration is that our method is confined to a grid, illustrated in Fig. 2.2. This has two drawbacks. Firstly, it can sample at most one feature per cell. Secondly, each cell is blind to its neighbours. Our method may thus select two contiguous pixels as distinct keypoints. At inference time we can work around this issue by applying non-maxima suppression on the feature map **K**, returning features at all local maxima. This addresses both issues at the cost of a misalignment between training and inference, which is potentially sub-optimal. We discuss this further in Sec. 2.5.4.

2.5 Experiments

We first describe our specific implementation and the training data we rely on. We then evaluate our approach on three different benchmarks, and present two ablation studies.

Training data. We use a subset of the MegaDepth dataset [124], from which we choose 135 scenes with 63k images in total. They are posed with COLMAP, a state-of-the-art SfM framework that also provides dense depth estimates we use to establish pixel-to-pixel correspondences. We omit scenes that overlap with the test data of the Image Matching Challenge (Sec. 2.5.1), and apply a simple co-visibility heuristic to sample viable pairs of images. See the supplementary material for details.

Feature extraction network. We use a variation of the U-Net [188] architecture. Our model has 4 down- and up-blocks which consist of a single convolutional layer with 5×5 kernels, unlike the standard U-Net that uses two convolutional layers per block. We use instance normalization instead of batch normalization, and PReLU non-linearities. Our models comprise 1.1M parameters, with a formal receptive field of 219×219 pixels.

Optimization. Although the matching stage has a single learnable parameter, θ_M, we found that gradually increasing it with a fixed schedule works well, leaving just the feature extraction network to be learned with gradient descent. Since the training signal comes from *matching features*, we process three co-visible images A, B and C per batch. We then evaluate the summation part of equation 2.1 for pairs $A \leftrightarrow B$, $A \leftrightarrow C$, $B \leftrightarrow C$ and accumulate the gradients w.r.t. θ. While matching is pair-wise, we obtain three image pairs per image triplet. By contrast, two pairs of unrelated scenes would require four images. Our approach provides more matches while reducing GPU memory for feature extraction. We rescale the images such that the longer edge has 768 pixels, and zero-pad the shorter edge to obtain a square input; otherwise we employ no data augmentation in our pipeline. Grid cells are square, with each side $h = 8$ pixels.

Rewards are $\lambda_{\text{tp}} = 1, \lambda_{\text{fp}} = -0.25$ and $\lambda_{\text{kp}} = -0.001$. Since a randomly initialized network tends to generate very poor matches, the quality of keypoints is negative on average at first, and the network would cease to sample them at all, reaching a local maximum reward of 0. To avoid that, we anneal λ_{fp} and λ_{kp} over the first 5 epochs, starting with 0 and linearly increasing to their full value at the end.

We use a batch of two scenes, with three images in each. Since our model uses instance normalization instead of batch normalization, it is also possible to accumulate gradients over multiple smaller batches, if GPU memory is a bottleneck. We use ADAM [106] with learning rate of 10^{-4}. To pick the best checkpoint, we evaluate performance in terms of pose estimation accuracy in stereo, with DEGENSAC [37]. Specifically, every 5k optimization steps we compute the mean Average Accuracy (mAA) at a 10^{o} error threshold, as in [95]: see Sec. 2.5.1 and the appendix for details.

Finally, our method produces a variable number of features. To compare it to others under a fixed feature budget, we subsample them by their "score", that is, the value of heatmap $\mathbf{K}$ at that location.

	Up to 2048 features/image							Up to 8000 features/image						
	Task 1: stereo			Task 2: Multiview				Task 1: stereo			Task 2: Multiview			
Method	NM	NI	**mAA(10^o)**	NM	NL	TL	**mAA(10^o)**	NM	NI	**mAA(10^o)**	NM	NL	TL	**mAA(10^o)**
Upright Root-SIFT	194.0	112.3	0.3986	199.3	1341.7	4.09	0.5623	525.4	358.9	0.5075	542.9	4404.6	4.38	0.6792
Upright L2-Net	174.1	117.1	0.4192	179.8	1361.3	4.23	0.5968	657.3	435.7	0.5450	395.5	3603.8	4.38	0.6849
Upright HardNet	274.0	152.7	**0.4609**	201.3	1467.9	4.31	0.6354	791.7	527.6	0.5728	509.1	4250.4	4.55	0.7231
Upright GeoDesc	235.8	132.7	0.4136	161.1	1287.3	4.24	0.5837	598.9	409.9	0.5267	458.6	4146.8	4.41	0.7044
Upright SOSNet	265.6	171.2	0.4505	194.0	1442.3	4.31	0.6359	752.9	508.4	0.5738	464.4	3988.6	4.52	0.7129
Upright LogPolarDesc	296.8	162.2	0.4567	211.9	1553.4	4.33	0.6370	821.7	543.2	0.5510	505.4	4414.1	4.52	0.7109
SuperPoint	292.8	126.8	0.2964	169.3	1184.3	4.34	0.5464	–	–	–	–	–	–	–
LF-Net	191.1	106.5	0.2344	196.7	1385.0	4.14	0.5141	–	–	–	–	–	–	–
D2-Net (SS)	505.7	188.4	0.1813	513.1	**2357.9**	3.39	0.3943	1258.2	482.3	0.2228	1278.7	5893.8	3.62	0.4598
D2-Net (MS)	327.8	134.8	0.1355	337.6	**2177.3**	3.01	0.3007	1028.6	470.6	0.2506	1054.7	**6759.3**	3.39	0.4751
R2D2	273.6	213.9	0.3346	280.8	1228.4	4.29	0.6149	1408.8	**842.2**	0.4437	739.8	4432.9	4.59	0.6832
Submission #609	439.7	**270.0**	**0.4690**	280.4	1489.6	**4.69**	**0.6812**	–	–	–	–	–	–	–
Submission #578	439.5	**246.6**	0.4542	331.6	1621.7	**4.57**	**0.6741**	–	–	–	–	–	–	–
Submission #599	227.4	129.5	0.4507	176.6	1209.6	4.44	0.6609	–	–	–	–	–	–	–
Submission #611	–	–	–	–	–	–	–	945.4	622.1	**0.5887**	899.1	6086.2	**4.65**	**0.7513**
Submission #613	–	–	–	–	–	–	–	934.9	**624.1**	**0.5873**	964.8	**6350.7**	4.64	**0.7495**
Submission #625	–	–	–	–	–	–	–	945.4	605.1	**0.5878**	899.1	6095.8	**4.65**	0.7485
DISK (#708 & #709)	514.2	**404.2**	**0.5132**	527.5	**2428.0**	**5.55**	**0.7271**	1621.9	**1238.5**	0.5585	1663.8	**7484.0**	**5.92**	**0.7502**
Δ (%)	*+1.7*	*+49.7*	*+9.4*	*+2.8*	*+3.0*	*+18.3*	*+6.7*	*+15.1*	*+47.1*	*-5.4*	*+30.1*	*+10.7*	*+27.3*	*-0.1*

Table 2.1: **Image Matching Challenge results.** The primary metric is **(mAA)**, the mean Average Accuracy in pose estimation, up to 10^o. We also report **(NM)** the number of matches (given to RANSAC for stereo, and to COLMAP for multiview). For stereo, we also report **(NI)** the number of RANSAC inliers. For multiview, we also report **(NL)** number of landmarks (3D points), and **(TL)** track length (observations per landmark). The top 3 results are highlighted in **red**, **green** and **blue**.

2.5.1 Evaluation on the 2020 Image Matching Challenge

The Image Matching Challenge (IMC) [95] provides a benchmark that can be used to evaluate local features for two tasks: stereo (Fig. 2.3) and multi-view reconstruction (Fig. 2.4). For the stereo task, features are extracted across every pair of images and then given to RANSAC, which is used to compute their relative pose. The multiview task uses COLMAP to generate SfM reconstructions from small subsets of 5, 10, and 25 images. The differentiating factor for this benchmark is that both tasks are evaluated *downstream*, in terms of the quality of the reconstructed poses, which are compared to the ground truth, by using the mean Average Accuracy (mAA) up to a 10-degree error threshold. While this requires carefully tuning components extraneous to local features, such as RANSAC hyperparameters, it measures performance on real problems, rather than intermediate metrics.

Hyperparameter selection. We rely on a validation set of two scenes: "Sacre Coeur" and "St. Peter's Square". We resize the images to 1024 pixels on the longest edge, generate cycle-consistent matches with the ratio test, with a threshold of 0.95. For stereo we use DEGENSAC [37], which outperforms vanilla RANSAC [95], with an inlier threshold of 0.75 pixels.

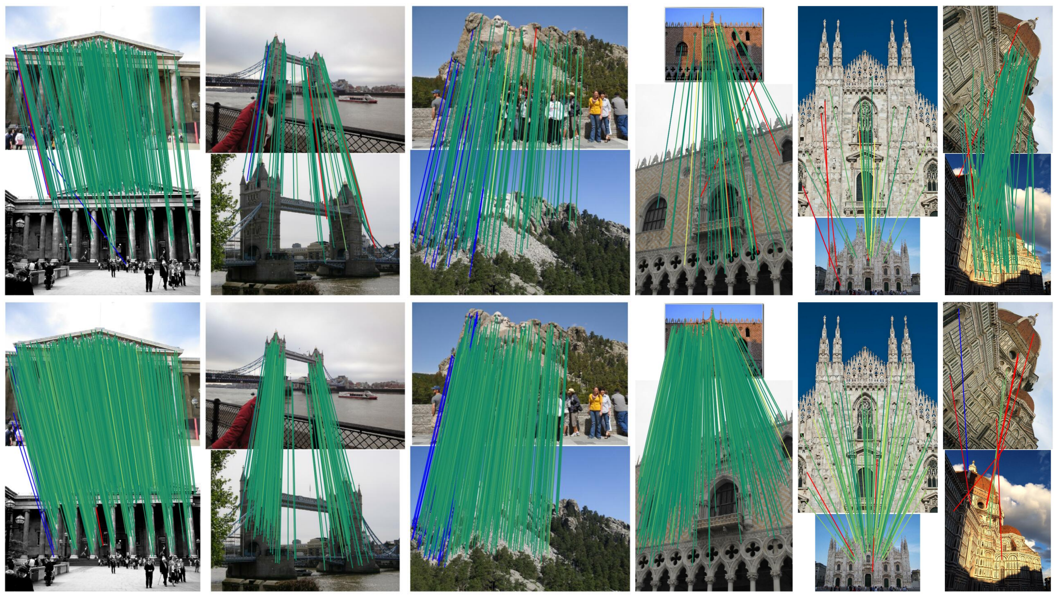


Figure 2.3: **Stereo results on the Image Matching Challenge (2k features).** Top: DoG w/ Upright HardNet descriptors [149]. Bottom: DISK. We extract cycle-consistent matches with optimal parameters and feed them to DEGENSAC [37]. We plot the resulting inliers, from green to yellow if they are correct (0 to 5 pixels in reprojection error), in red if they are incorrect (above 5), and in blue if ground truth depth is not available. Our approach can match many more points and produce more accurate poses. It can deal with large changes in scale (4th and 5th columns) but not in rotation (6th column), which is discussed further in section 2.5.1 and Fig. 2.7.

Results. We extract DISK features for the nine test scenes, for which the ground truth is kept private, and submit them to the organizers for processing. The challenge has two categories: up to 2k or 8k features per image. We participate in both. We report the results in Tab. 2.1, along with baselines taken directly from the leaderboards, computed in [95]. We consider several descriptors on DoG keypoints: RootSIFT [133, 5] L2-Net [217], HardNet [149], GeoDesc [137], SOSNet [218] and LogPolarDesc [54]. For brevity, we show only their upright variants, which perform better than their rotation-sentitive counterparts on this dataset. For end-to-end methods, we consider SuperPoint [46], LF-Net [161], D2-Net [52], and R2D2 [183]. All of these methods use DEGENSAC [37] as a RANSAC variant for stereo, with their optimal hyperparameters. We also list the top 3 user submissions for each category, taken from the leaderboards on June 5, 2020 (the challenge concluded on May 31, 2020).

On the 2k category, we outperform all methods by 9.4% relative in stereo, and 6.7% relative in multiview. On the 8k category, averaging stereo and multiview, we outperform all baselines, but place slightly below the top three submissions. Our method can find many more matches than any other, easily producing 2-3x the number of RANSAC inliers or 3D landmarks. Our features used for the 2k category are a subset of those used for 8k, which indicates a potentially sub-optimal use of the increased budget, which may be solved training with larger images or smaller grid cells. We show qualitative images in Figs. 2.3 and 2.4.

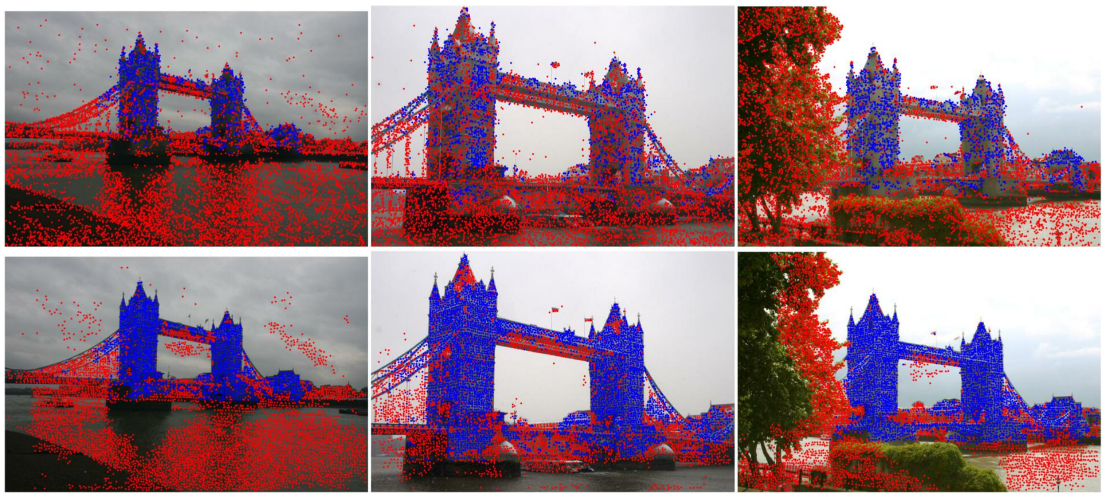
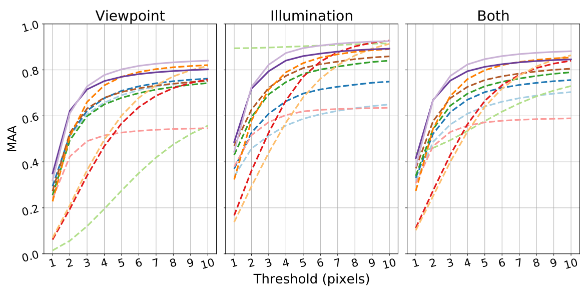


Figure 2.4: **Multiview results on the Image Matching Challenge (8k features).** Top: DoG w/ Upright HardNet descriptors [149]. Bottom: DISK. COLMAP is used to reconstruct the "London Bridge" scene with 25 images. We show three of them and draw their keypoints, in blue if they are registered by COLMAP, and red otherwise. Our method generates evenly distributed features, producing 76% more landmarks with 30% more observations per landmark than HardNet. Keypoints on water or trees have low scores and are rare among the top 2k features, but appear more often when taking 8k. This suggests that our method can reach near-optimal performance on a small budget.

Note that we only compare with submissions using the built-in feature matcher, based on the l_2 distance between descriptors, instead of neural-network based matchers [239, 246, 198], which combined with state-of-the-art features obtain the best overall results. Even so, DISK places #2 below only SuperGlue [198] on the 2k category, outperforming *all other solutions* using learned matchers.

Rotation invariance. We observe our models break under large in-plane rotations, which is to be expected. We evaluate their performance with an additional test using synthetic data. We pick 36 images randomly from the IMC 2020 validation set, match them with their copies, rotated by θ, and calculate the ratio of correct matches, defined as those below a 3-pixel reprojection threshold. In Fig. 2.7 we report it for different state-of-the-art methods that, like ours, bypass orientation detection, and overlay a histogram of the differences in in-plane rotation in the dataset. We find that DISK is exceptionally robust to the range of rotations it was exposed to, and loses performance outside of this range, suggesting that failure modes such as in Fig. 2.3 can be remedied with data augmentation.



Method	Num. Features	Num. Matches	Viewp. AUC(5px)	Illum. AUC(5px)	Both AUC(5px)
HesAff/RootSIFT	6710.1	2851.7	0.552	0.491	0.523
HAN/HardNet++	3860.8	1960.0	0.564	0.573	0.568
DELF	4590.0	1940.3	0.132	**0.898**	0.501
SuperPoint	1562.6	883.4	0.535	0.650	0.590
Reinforced FP	—	—	0.563	0.680	0.621
LF-Net	500.0	177.5	0.439	0.538	0.487
D2-Net (SS)	5965.1	2495.9	0.326	0.499	0.409
D2-Net (MS)	8254.5	2831.6	0.349	0.424	0.385
R2D2	4989.8	1846.4	0.580	0.647	0.613
DISK (2k)	2048.0	1024.2	0.642	0.740	0.689
DISK (8k)	7705.1	3851.8	**0.648**	0.752	**0.698**

Figure 2.5: **Results on HPatches.** On the left, we report Mean Matching Accuracy (MMA) at 10 pixel thresholds. On the right, we summarize MMA by its AUC, up to 5 pixels. Results for RFP [17] were kindly provided by the authors, which explains why keypoint/match counts are missing.

2.5.2 Evaluation on HPatches

HPatches [10] contains 116 scenes with 6 images each. These scenes are strictly planar, containing only viewpoint or illumination changes (not both), and use homographies as ground truth. Despite its limitations, it is often used to evaluate low-level matching accuracy. We follow the evaluation methodology and source code from [52]. The first image on every scene is matched to the remaining five, omitting 8 scenes with high-resolution images. Cyclic-consistent matches are computed, and performance is measured in terms of the Mean Matching Accuracy (MMA), *i.e.*, the ratio of matches with a reprojection error below a threshold, from 1 to 10 pixels, and averaged across all image pairs.

We report MMA in Fig. 2.5, and summarize it by its Area under the Curve (AUC), up to 5 pixels. Baselines include RootSIFT [133, 5] on Hessian-Affine keypoints [146], a learned affine region detector (HAN) [150] paired with HardNet++ descriptors [149], DELF [160], SuperPoint [46], D2-Net [52], R2D2 [183], and Reinforced Feature Points (RFP) [17]. For D2-Net we include both single- (SS) and multi-scale (MS) models. We consider DISK with number of matches restricted to 2k and 8k, for a fair comparison with different methods.

We obtain state-of-the-art performance on this dataset, despite the fact that our models are trained on non-planar data without strong affine transformations. We use the same models and hyperparameters used in the previous section to obtain 2k and 8k features, without any tuning. Our method is #1 on the viewpoint scenes, followed by R2D2, and #2 on the illumination scenes, trailing DELF. Putting them together, it outperforms its closest competitor, RFP, by 12% relative.

2.5.3 Evaluation on the ETH-COLMAP benchmark

The ETH-COLMAP benchmark [203] compiles statistics for large-scale SfM. We select three of the smaller scenes and report results in Tab. 2.6. Baselines are taken from [17] and include Root-SIFT [133, 5], SuperPoint [46], and Reinforced Feature Points [17]. We obtain more landmarks than SIFT, with larger tracks and a comparable reprojection error. Note that this benchmark does

Scene	Method	NL	TL	ϵ_r
Fountain	Root-SIFT	15k	4.70	**0.41**
	SP	**31k**	4.75	0.97
	RFP	9k	4.86	0.87
	DISK	18k	**5.52**	0.50
Herzjesu	Root-SIFT	8k	4.22	**0.46**
	SP	**21k**	4.10	0.95
	RFP	7k	4.32	0.82
	DISK	11k	**4.71**	0.48
South Building	Root-SIFT	113k	5.92	**0.58**
	SP	**160k**	7.83	0.92
	RFP	102k	7.86	0.88
	DISK	115k	**9.91**	0.59

Figure 2.6: **Results on ETH-COLMAP [203]**. We compare Root-SIFT [133], SuperPoint [46], Reinforced Feature Points [17], and DISK. We report: **(NL)** number of landmarks, **(TL)** track length (average number of observations per landmark), and (ϵ_r) reprojection error.

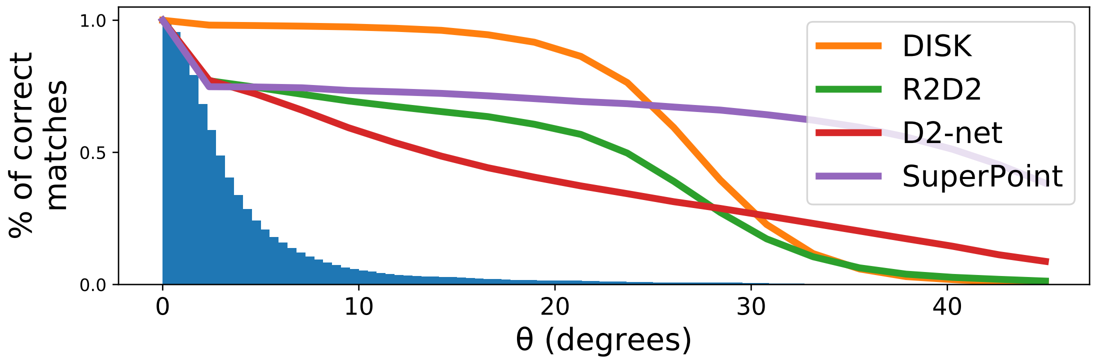


Figure 2.7: **Rotation invariance vs. rotations in data.** We report the ratio of correct matches between a reference images and their copies rotated by θ. Overlaid is a histogram of relative image rotations in IMC2020-val.

	2k features		8k features	
Variant	Stereo	Multiview	Stereo	Multiview
Depth	**0.7218**	0.8325	**0.7767**	0.8628
Epipolar	0.7145	**0.8465**	0.7718	**0.8749**

Figure 2.8: **Ablation: match supervision.** We compare mAA on the Image Matching Challenge validation set, for DISK models learned with pixel-to-pixel supervision or epipolar constraints.

Variant	Num. features	Num. matches	Stereo mAA(10^0)	Multiview mAA(10^0)
1-per-cell	5456.8	796.5	0.74774	0.84685
NMS 3×3	8434.6	1699.9	**0.77833**	0.86864
NMS 5×5	7656.0	1547.9	0.77657	**0.87622**
NMS 7×7	6423.4	1271.1	0.77070	0.85642
NMS 9×9	4946.2	942.0	0.75558	0.85362

Figure 2.9: **Ablation: NMS.** We compare the feature selection strategy used for training (top) with NMS at inference time. Here we use *all detected features*, rather than subsample by score.

Grid \ NMS	3×3	5×5	7×7	9×9
8×8	0.7751	**0.7824**	0.7778	0.7586
12×12	0.7576	**0.7580**	0.7502	0.7431
16×16	0.7213	**0.7214**	0.7120	0.6999

Figure 2.10: **Ablation: NMS vs grid size**. We show mAA vs. grid & NMS size on IMC2020-val, capping the number of features to 2k.

not standardize the number of input features, so we extract DISK at full resolution and take the top ~12k keypoints in order to remain comparable with SIFT. By comparison, a run on “Fountain” with no cap yields 67k landmarks.

2.5.4 Ablation studies and discussion

Supervision without depth. As outlined in Sec. 2.4, we use the strongest supervision signal available to us, which are depth maps. Unfortunately, this means we only reward matches on areas with reliable depth estimates, which may cause biases. We also experimented with a variant of R that relies only on *epipolar constraints*, as in a recent paper [228]. We evaluate both variants on the validation set of the Image Matching Challenge and report the results in Tab. 2.8. Performance improves for multiview but decreases for stereo. Qualitatively, we observe that new keypoints appear on textureless areas outside object boundaries, probably due to the U-Net’s large receptive field (see appendix). Nevertheless, this illustrates that DISK can be learned just as effectively with much weaker supervision.

Non-maximum suppression and grid size. The softmax-within-grid training time mechanism models the relative importance of features under a constrained budget, in a differentiable way. It can be replaced with an alternative solution, such as NMS, which we use at inference. In Tab. 2.9 we compare the training regime, where we sample at most one feature per grid cell, against the inference regime, where we apply NMS on the heatmap. We report results in terms of pose mAA on the validation set of the Image Matching Challenge in Tab. 2.9. For this experiment we removed the budget limit and took all features provided by the model. This shows that this inference strategy is clearly beneficial, despite departing from the training pipeline. In Tab. 2.10 we show how mAA varies with grid size used for training. A smaller grid is beneficial in terms of performance but increases the number of extracted features, leading to larger distance matrices and higher computational expense.

Feature duplication at grid edges. Experimentally, we observe that 19.9% of features from grid selection (training) have a neighbour within 2 px, which likely corresponds to double detections. This has three potential downsides. (1) Compute/memory is increased, due to unnecessarily large matching matrices. (2) It rescales λ_{kp} w.r.t. its intuitive meaning. Imagine that some detections are *strictly duplicated*: both forward and backward probabilities will "split in half", but the total probability of matching the two locations remains constant – this means that learning dynamics are not impacted, other than λ_{kp} acting more strongly (on a larger number of detections). (3) In reality, detections are *close by*, instead of duplicated, which may make the algorithm less spatially precise: since duplication means a failure of the sparsity mechanism, we learn in a regime where imprecise correspondences are more common than at inference, favoring shift-invariance in the descriptors more than desired. The results DISK attains on HPatches, including at a 1-pixel error threshold, and the very low reprojection error on the ETH-COLMAP benchmark, suggest that these do not pose a significant problem for performance.

2.6 Project background

The project which resulted in DISK started around October 2019. At that time, prior work on detection and description learned the two simultaneously but without a two-way interdependency between the keypoints and descriptors. This means that the two tasks were solved by a single network, mostly for inference speed and to benefit from multi-task regularization, but not leading to co-adaptation of keypoints and descriptors. The motivation for DISK was to recognize the setting as a discrete decision problem and solve it as a whole, in a principled way. Compared to the prior work this meant replacing their multiple additive loss terms with a single, easily interpretable reward function. This reward was to be maximized, while otherwise imposing minimal constraints and initialization biases, allowing the network to find the best possible solution. Finally, due to the first edition of the Image Matching Challenge [95], we aimed at evaluating our algorithm in terms of the downstream task of camera pose estimation, compared to prior work which used match and keypoint quality metrics like repeatability and accuracy on small and often biased datasets like Brown [21] or HPatches [10].

Optimizing for pose recovery. DISK, as presented in NeurIPS 2020, differed from the initial goals in two major ways. Firstly, we initially aimed to supervise based on stereo pose estimation error, which is our ultimate quality metric. That idea was abandoned when at the beginning of December 2020 Bhowmik et al. published Reinforced Feature Points (RFP) [17] which attempted just that. We suspected that the minor performance gain they obtained over the pretrained SuperPoint network was due to the credit assignment problem. Compared to assigning each keypoint an individual reward depending on whether it was matched correctly, with RFP all keypoints get the same reward based on the pose error of the entire set of matches. This means that to establish which keypoints and matches perform well, the practitioner has to empirically marginalize the reward over all possible subsets of keypoints and matches, leading to very high variance in the gradient signal. We therefore decided to stick with a proxy for this goal and benefit from precise gradient estimates.

Training-time sparsity control. Secondly, at the start of the project we believed that precisely controlling the sparsity of keypoints would be an important part of the solution. This motivated introducing a mechanism similar to non-maxima suppression (NMS) at training time. The network forward pass would produce an *initial* keypoint heatmap. It would then be divided into interleaved sectors and sampled from, sector-by-sector, while updating heatmap values in each samples' neighbourhood. The detail is shown in Fig. 2.11. Towards the end of the project we found that this module did not contribute to improved network performance. We removed this mechanism in favor of simply picking on point per grid cell. This was sufficient for training, yielding superior downstream results when used with classic NMS for inference. A similar idea was later presented in S-TREK [197] which shows that with careful implementation modelling NMS at training time can be beneficial after all.

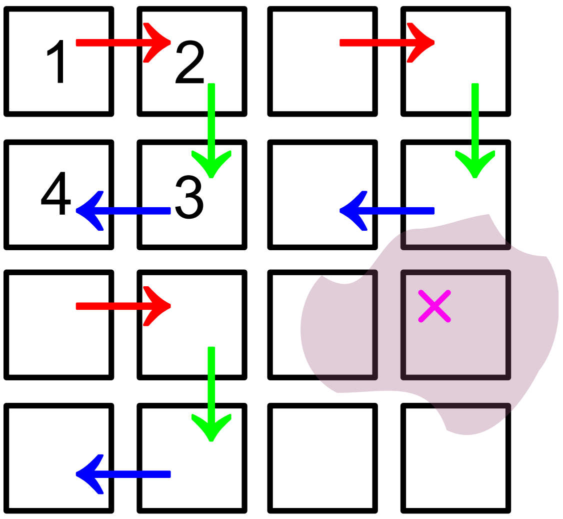


Figure 2.11: The original mechanism for maintaining keypoint sparsity in DISK. The image is divided into a grid of cells, which are assigned to groups 1-4. In rounds $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, a single keypoint is sampled from each cell in the group. After each round, a local update to the heatmap is made to reduce the probability of sampling future keypoints in the immediate neighborhood. With the update size up to $2\times$ grid size (magenta), points in each group can be sampled simultaneously without overlapping the updates.

Close to the time of submission of DISK (June 2020), the Image Matching Challenge 2020 [95] showed that much of the prior work on learned keypoint *detectors* was in fact not improving in downstream performance over handcrafted alternatives paired with a well tuned learned descriptor. In particular the Difference of Gaussians (DoG) detector paired with HardNet [149] proved very effective. This set the stage for DISK, which substantially improved on that baseline only to be overshadowed by the emergence of learned feature *matchers*, in particular SuperGlue [198].

2.7 Impact

Since its publication in 2020, DISK has been cited over 175 times and garnered interest of the community, being used as a building block in the subsequent editions of the Image Matching Challenge, including the winner of the 2021 edition. DISK has also received interest from the industry, with us being aware of applications in aerial image stitching and internal trials at Google. It is also integrated in Kornia [185], a popular library of computer vision algorithms, PySLAM v2 [61], a popular visual odometry package built for educational purposes. It is also one of the available feature backends in LightGlue [127], an open source and improved variant of SuperGlue [198].

One of the most influential ideas presented in DISK is the double-softmax relaxation of nearest neighbour matching. The idea is directly credited to DISK in [68] and [213] although the latter mentions its first introduction in [186], of which we were not aware. Otherwise most following works explore approaches different to our policy gradient formulation [11, 248, 68, 55]. Notably [197] reuses our approach for training a *repeatable* keypoint detector while deferring the learning of descriptors to later stage of training.

2.8 Future work

Despite the interest from the community mentioned above and the effort put in reproducibility of the training procedure, we are not aware of any openly available retrainings of DISK with different data or settings. This is perhaps due to our lack of transparency about the abundance of remaining low hanging fruit, as the algorithm was published with very limited hyperparameter tuning and data engineering. In this section we discuss these areas for improvement in hope that it will be useful to the community wishing to adapt the model to their needs.

2.8.1 Network architecture

There are inherent tradeoffs in the choice of network architecture, in particular speed versus expressivity, model size and memory consumption. Still, the model is not at the Pareto frontier and direct improvements can be made without sacrificing other aspects of model performance.

Normalization layers. The U-Net in DISK uses Instance Normalization (IN) layers [224], which normalize each channel of a 2D tensor to $\mu = 0, \sigma = 1$ independently across the batch dimension b. This is in contrast to the more popular Batch Normalization (BN) [87], which also normalizes per-channel but across all items in a batch. The reason for this choice is the small b used in DISK, due to GPU memory constraints. In the limit of small b, BN has two downsides. Firstly, it may introduce excessive stochasticity in the network's internal representations because the estimates of μ, σ are computed over a small sample. Secondly, specifically to DISK, computing the average across both images in a pair introduces a backdoor channel for information flow between the two. With large b, the influence of the paired image on μ and σ is negligible compared to all other images. However in our setting, with $b = 3$ image pairs, DISK could silently exploit this information flow which will not be available at inference time.

For this reason we opted for IN as an alternative which works on images individually, solving both problems. At the same time, IN is known to sometimes be too aggressive, "washing out" too much information from the feature maps and causing issues with retaining it across layers. An intermediate solution is Group Normalization [233] which divides feature channels into arbitrary groups of a given size and normalizes over those jointly, while keeping batch elements separate. For future work, we suggest that a well tuned application of GN is likely to perform better than the current IN. Alternatively BN can bring inference runtime benefits if we can train the model with a larger batch size b, as discussed in Sec. 2.8.2.

Finally, we point out that the choice of normalization layers may be much more important for local feature detection and description than for most other tasks. Computing μ, σ over the entire image effectively results in *global receptive field*, which the network can use to suppress the detection of patterns which are repeating in the specific image. Depending on the extent to which this mechanism is leveraged by the network, the choice of BN could unexpectly harm performance.

Convolutional blocks. DISK uses a very simple U-Net structure where each block consists of a normalization layer (without learnable scale and bias), a PReLU activation, and a 5×5 2D convolution. The simplicity of this implementation does not necessarily translate to parameter or runtime efficiency though. Later works [81, 131] show a superior tradeoff in that respect by separating spatial and channel-wise operation of the kernels. We expect these results apply to DISK as well, suggesting possible improvements to runtime and parameter counts at equivalent performance. A downside of space-depth factorized convolutions is the extra training-time memory cost of storing their intermediate representations. We discuss this in Sec. 2.8.2. Also, even though the nonlinearity is “leaky”, enabling scale and bias on the preceding normalization layer may be beneficial.

Reducing the computational cost of the network. We chose U-Net as a backbone because of its proven capability in high-resolution image processing regimes, but its naive application is not the most efficient approach to the task. The biggest downside is having to compute a tensor of high dimensional descriptors for each pixel in the image, to only use individual locations with sparsity on the order $\frac{1}{64}$. This is wasteful both in terms of memory and computation. If the keypoints were chosen first, a sparse convolution can be used to evaluate the descriptors only at those locations, for large savings. Unfortunately, sparse convolutions are not supported by PyTorch and require third-party extensions, causing engineering overhead. An alternatve approach is that of SuperPoint [46] which uses bilinear interpolation to obtain per-pixel descriptors from a lower resolution tensor. This is less expressive than a full-resolution convolution but is a tradeoff which may be worth taking for many applications. Another alternative proposed in [248] is a standard CNN backbone used to create a feature scale pyramid. The descriptors are formed by concatenating the results of a bilinear lookup across all resolutions.

Hyperparameter search. We conducted no substantial hyperparameter search with DISK. This means that changes such as the optimization algorithm, learning rate, weight decay (not used), network depth/width tradeoffs, batch size, reward settings are all likely to lead to improved performance.

2.8.2 Data and training procedure

The second and likely larger area for improvements is with data engineering. Similarly as with architecture, some changes are likely to result in general performance improvements while others

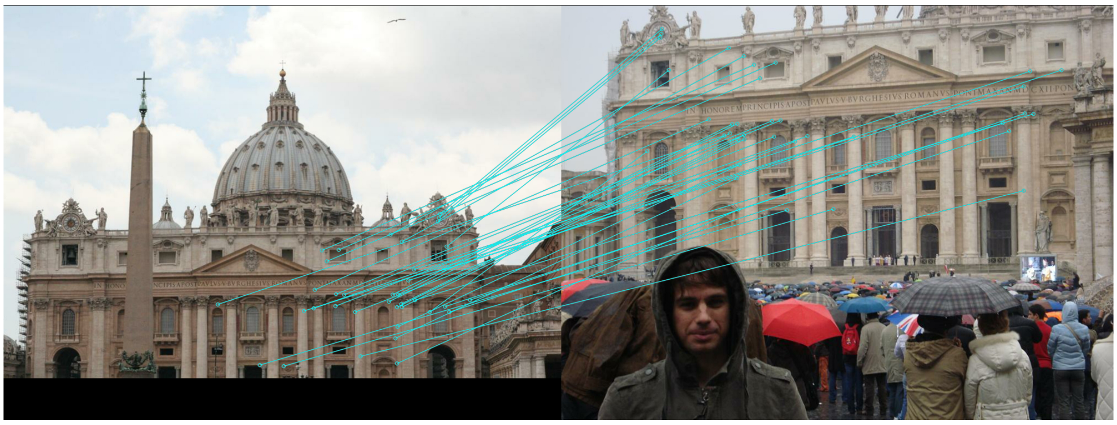


Figure 2.12: A partially symmetric scene in **left** view has been cropped in the **right** view. *Incorrect* matches are marked in cyan. The dataset is biased towards full, object-centric views and such pairs are rare. Removing this bias could lead to better use of context for description, a capability demonstrated in Fig. 1.1.

are application-specific.

Non object-centric data. Common applications of feature matching are photogrammetry and SLAM, where the images are captured to cover the scene, rather than to depict a specific object. This is in contrast with the MegaDepth [124] dataset DISK was trained on, which in majority consists of handheld photos of tourist landmarks. These are largely centered on that landmark, with most images specifically choosing the most picturesque angles. Overall this leads to a bias in the dataset, which is especially pronounced on non object-centric data, but can be observed even in distribution. Fig. 2.12 illustrates a failure mode when a partially symmetric building has been cropped in one of the views, resulting in feature mismatches. This happened even though DISK is capable of encoding non-local information in feature descriptors, such as distinguishing the left and right sides of buildings in Fig. 1.1. While general and robust treatment of symmetric structures is an active area of research [23] and beyond the scope of local features, we argue that the problem can be reduced by addressing the data bias — the simplest solution to which is expanding the dataset. This can include custom data capture or datasets already available for other purposes, such as autonomous driving [64, 40, 58] or indoor scans [42, 243]. These potential inclusions come with their own biases, such as lower variation in lighting and weather conditions or large numbers of dynamic objects. For this reason, below we suggest approaches which rely on the MegaDepth dataset itself.

Data augmentation. The simplest approach to tackling object-centricity of the dataset is data augmentation, specifically image cropping. MegaDepth is a mix of portrait and landscape pictures, which is problematic when trying to unify their shape for batch processing. Currently, DISK deals with non-square images by padding them with black pixels. The alternative is to crop a random square subregion. Our choice was to avoid pairs with low covisibility which may become entirely disjoint after a crop and cause potential learning issues. From today's perspective we believe random cropping may be an overall better choice. It may also help to consider more general geometric transformations, such as skews and displacement grids, as long as they can be inverted for evaluating match correctness. We are less optimistic about photometric data augmentation, as the variability of lighting and weather conditions is already one of the biggest strengths of MegaDepth, thanks to its crowd-sourced nature. Still, powerful detectors and descriptors have been trained solely with image augmentation [68, 46] so further performance improvements cannot be ruled out.

More difficult image pairs. A unique aspect of image matching is working with image pairs. When used for its original purpose of monocular depth, MegaDepth with its 63k images is a modestly sized dataset. However, since the number of image pairs grows quadratically with scene size, just the Notre Dame scene comprises of over 1 million pairs. Although this means that overfitting in this setting is less likely, not all pairs are made equal. To pick the pairs for training DISK we took the set of 3D (SIFT) landmarks registered with each image $\{L_a\}, \{L_b\}$ and used their

$$\text{IoU}(a,b) = \frac{|\{L_a\} \cap \{L_b\}|}{|\{L_a\} \cup \{L_b\}|} \tag{2.2}$$

as a proxy for covisibility, selecting pairs in the range of $0.9 - 0.3$. Not starting at 1 is meant to exclude near-duplicate images but given SIFT's relatively low matching accuracy it may still lead to many overly simple image pairs. Since DISK is not optimized for the downstream task of geometry estimation but rather for the number of correct matches, this may lead to the network focusing on making unnecessarily many matches in easy image pairs while not providing enough for the difficult ones. This can be addressed by more appropriate choice of pairs, for example lowering the IoU threshold to focus more on difficult cases.

Bootstrapping the dataset. A more radical approach would be use DISK as is, which already offers substantial performance benefits over SIFT, to rebuild the pseudo-ground truth SfM models used for training. Despite the recent progress in local features and image matching in general, the most widely dataset is MegaDepth, which uses RootSIFT[5] matches for SfM. Updating the datasets would allow for better camera poses and for calculating more representative image pair IoU (see the previous paragraph), thanks to DISK's substantially higher number of observations per landmark (Fig. 2.1). This will be especially pronounced on scenes with fewer images. These new scene models can be used directly or compared with the RootSIFT ones to filter out problematic images. We also remark that the "Reichstag" scene of the IMC-2020 validation set seems to contain mis-posed images, which may be especially impactful for final DISK performance as it is used for early stopping (along with only two others).

Memory-efficient training. A large contributor to the complexity of DISK implementation is the struggle with GPU memory limitations. The main culprits are a) the full resolution descriptor maps computed by the model and b) descriptor distance matrices computed in the loss function. For the latter we employ a trick where we compute the matrices one by one, backpropagating the loss after each image pair to immediately free up the used memory and re-use it for the next image pair. As for a), at 768×768 resolution and with 128-dimensional 32-bit descriptors, storing them for one image requires 302MB of memory and puts an upper bound on batch size of $b \approx 100$ with a 32GB GPU. At the same time, saving memory allows for larger b, which can lead to more stable optimization and enables batch normalization and space-depth factorized convolutions mentioned in Sec. 2.8.1. For this reason improving the memory efficiency of the training pipeline is an important direction. One way is through activation checkpointing the U-Net and the other is using reduced precision training schemes. The latter needs to account for high activation values and gradients of the keypoint heatmap, which may lead to numerical overflows. Finally, adapting the code to run on multiple GPUs would also enable experimenting with more expensive features.

2.8.3 Variants

As indicated in the previous subsections, many design choices for DISK represent tradeoffs which can only be evaluated in the context of a specific application. It is reasonable to prepare a number of models optimized for different usecases:

- A large and strong baseline model. Optimized for high resolution images, high feature counts (8k+), with a large network. For use in GPU-enabled settings for stereo with difficult image pairs and/or obtaining high density sparse SfM models.
- A medium model for efficient SfM pose estimation, optimized for medium-high resolution images with ~ 4k feature detections, with 64 dimensional descriptors. For inference with medium image collections on CPU or for very large image collections on GPU.
- A small and fast model, optimized for edge, SLAM and robotics applications. Optimized for 512×512 or smaller images, aiming at 1-2k feature detections, with lower descriptor dimensionality (64 or 32). The network should be optimized for small compute and memory footprint, possibly with batch normalization to avoid reduction operations at inference time. Due to the nature of semi-sequential matching it can be trained on easier image pairs than the previous two variants.

We believe that these three variants reasonably cover the space of common applications of local features without a combinatorial explosion of versions.

3 RayTran

“RayTran: 3D pose estimation and shape reconstruction of multiple objects from videos with ray-traced transformers” by Michał Tyszkiewicz, Kevis-Kokitsi Maninis, Stefan Popov and Vittorio Ferrari in ECCV2022. Candidate’s contributions: most of system design (see Sec. 3.7), implementation.

3.1 Abstract

We propose a transformer-based neural network architecture for multi-object 3D reconstruction from RGB videos. It relies on two alternative ways to represent its knowledge: as a global 3D grid of features and an array of view-specific 2D grids. We progressively exchange information between the two with a dedicated bidirectional attention mechanism. We exploit knowledge about the image formation process to significantly sparsify the attention weight matrix, making our architecture feasible on current hardware, both in terms of memory and computation. We attach a DETR-style head [25] on top of the 3D feature grid in order to detect the objects in the scene and to predict their 3D pose and 3D shape. Compared to previous methods, our architecture is single stage, end-to-end trainable, and it can reason holistically about a scene from multiple video frames without needing a brittle tracking step. We evaluate our method on the challenging Scan2CAD dataset [8], where we outperform (1) state-of-the-art methods [141, 121, 120, 43] for 3D object pose estimation from RGB videos; and (2) a strong alternative method combining Multi-View Stereo [53] with RGB-D CAD alignment [7].

3.2 Introduction

Detecting and reconstructing objects in 3D is a challenging task with multiple applications in computer vision, robotics, and AR/VR that require semantic 3D understanding of the world. In this paper we propose RayTran, a transformer-based [225] neural network architecture for reconstructing multiple objects in 3D given an RGB video as input. Our key new element is a backbone which infers a global representation of the 3D volume of the scene. We attach a DETR-style head [25] on top of it, which detects objects in the 3D representation and predicts their 3D pose and shape (Figure 3.1).

The backbone inputs multiple video frames showing different views of the same static scene. Its task is to jointly analyze all views and to consolidate the extracted information into a global 3D

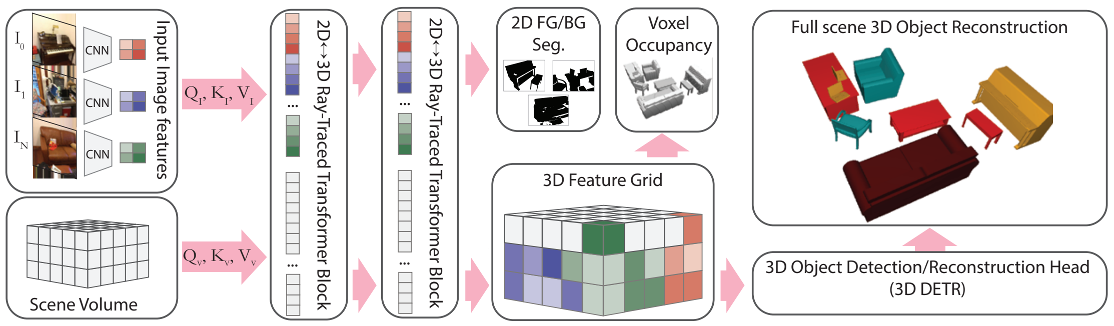

I0
I1
IN
CNN
Input Image features
Scene Volume
QI, KI, VI
QV, KV, VV
2D↔3D Ray-Traced Transformer Block
2D↔3D Ray-Traced Transformer Block
2D FG/BG Seg.
Voxel Occupancy
3D Feature Grid
Full scene 3D Object Reconstruction
3D Object Detection/Reconstruction Head (3D DETR)

Figure 3.1: **Overview of our method:** The RayTran backbone processes information in two parallel network streams. The first one (2D) works on features extracted on the multiple input frames. The second one (3D) starts from an empty volumetric feature representation of the scene. The 2D stream gradually consolidates features on the 3D volume and vice-versa with repeated blocks of ray-traced transformers. The backbone outputs a 3D feature grid which offers a global representation of the 3D volume of the scene. We attach a DETR-style head [25] to this representation, to detect all objects in 3D and to predict their 3D pose and 3D shape. We further help training with two auxiliary tasks: predicting 3D coarse binary occupancy for all objects together, and predicting amodal 2D foreground-background masks.

representation. Internally, the backbone maintains two alternative scene representations. The first is three-dimensional and describes the volume of the scene. The second is two-dimensional and describes the volume from the perspective of the individual views. We connect these two representations with a bidirectional attention mechanism to exchange information between them, allowing the 3D representation to progressively accumulate view-specific features, while at the same time the 2D representation accumulates global 3D features.

Processing videos with transformers is notoriously resource-consuming [15, 6]. Our case is no exception: if we relied on attention between all elements in the 2D and 3D representations, the attention matrix would have infeasible memory requirements (and it would also be computationally very expensive). To overcome this, we propose a *sparse ray-traced attention* mechanism. Given the camera parameters for each view, we exploit the image formation process to identify pairs of 2D and 3D elements that are unlikely to interact. We omit these pairs and store the attention matrix in a sparse format. This greatly reduces its computational and memory complexity, by a factor of $O(|V|^{\frac{2}{3}})$, where $|V|$ is the number of voxels in the 3D representation.

We attach a DETR-style head [25] on top of the 3D representation produced by the backbone. This head detects objects and predicts their class, 3D shape, and 3D pose (translation, rotation, scale). We represent object shapes with a voxel grid and then extract meshes using marching cubes [118]. We also predict coarse binary volumetric occupancy for all objects together, using a 3D convolutional layer on top of the global 3D representation. This provides an auxiliary task that teaches the network about the scene's geometry, and is essential for training.

As a second auxiliary task, we add an additional network head that predicts the 2D amodal foreground-background binary masks of all objects in the scene. Besides enabling this task, this head also helps training the backbone as it closes the loop between images and the 3D representation.

Several recent works [141, 190, 121] tackle 3D scene reconstruction from videos in the same setting. They rely on a 3-step pipeline: (1) object detection in individual 2D frames, along with estimating properties such as 3D rotation, parts of 3D scale, and 3D shape (either as a parametric surface [121] or by retrieving a CAD model from a database [141]); (2) tracking-by-detection [4, 20, 14], to associate 2D detections across frames; (3) multi-view optimization to integrate the per-frame predictions. This completes all 3D pose parameters, resolving the scale-depth ambiguities, and places all objects in a common, global 3D coordinate frame.

Our method was inspired by these works and addresses several of their shortcomings. The pipelines are composed of heterogeneous steps, which are trained separately and require manual tuning to work well together. The pipelines are complicated and over-engineered due to the intricate nature of the full-scene object reconstruction task. The tracking step is especially brittle. Objects often go out of view and re-appear later, and occlude each other over time. This poses a major challenge and leads to objects broken into multiple tracks, as well as tracks mixing multiple objects. These tracking errors harm the quality of the final 3D reconstructions.

In contrast, our method is end-to-end trainable. It is built from well understood neural network modules and it has a simple, modular architecture in comparison. Importantly, *we avoid tracking altogether*. Furthermore, our method does not rely on any notion of time sequence, so it is also applicable to sparse multi-view inputs (in addition to video).

We evaluate RayTran on the challenging Scan2CAD [8] dataset, featuring videos of complex indoor scenes with multiple objects. Through extensive comparisons we show that RayTran outperforms several works: (1) two baselines that process frames individually, defined in [141] as extensions of Mask2CAD [111]. This illustrates the value of jointly processing multiple frames in RayTran; (2) four recent multi-frame methods Vid2CAD [141], ODAM [121], MOLTR [120], ImVoxNet [43]. Besides performing better, RayTran also offers a much simpler design than [141, 121, 120], with an end-to-end trainable, unified architecture which does not require a tracking module; (3) a strong alternative method that combines the state-of-the-art Multi-View Stereo [53] and RGB-D CAD alignment [7] methods.

3.3 Related Work

3D from multiple views. Classic SfM/SLAM works cast 3D reconstruction as estimation of 3D points from multiple views based on keypoint correspondences [165, 155, 230, 202, 56]. However, the output point cloud is not organized into objects instances with their classes, 3D shapes, or poses. A line of works detect and localize objects in 3D using multi-view projection constraints, by approximating the object shapes with 3D boxes [237] and ellipsoids [158]. ODAM [121] goes a step further to creates a scene representation out of superquadrics, by using a graph neural network as core architecture for object association in time. FroDO [190] and MO-LTR [120] rely on both 2D image cues and the sparse 3D point clouds from SfM/SLAM to reconstruct objects in the scene. Qian et al. [174] produce volumetric reconstructions of multiple objects in a synthetically generated scene. Vid2CAD [141] integrates the single-view predictions of Mask2CAD [111] across time, to place objects from a CAD database into the 3D scene.

A common caveat of multi-view methods for 3D object reconstruction is that their architectures are overly complex, they cannot be trained end-to-end due to their heterogeneity, and they often rely on a brittle tracking-by-detection step. Instead, our proposed method provides a light-weight end-to-end architecture for the task, while we completely avoid tracking.

Similar to RayTran, the concurrent ImVoxelNet [43] keeps its 3D knowledge in a global 3D representation and does not require tracking. It uses a hand-crafted unidirectional mechanism to project and consolidate image features onto it. In contrast, our ray-traced transformers learn the optimal way to consolidate features. They are also bidirectional, which enables 2D supervision through re-projection as well as additional tasks, like novel-view synthesis. Moreover, RayTran reconstructs the 3D shapes of the detected objects, going beyond detecting 3D boxes.

Transformer architectures for computer vision. Several recent works use attention-based architectures (transformers) [225] for computer vision tasks. ViT [51] replaces the traditional convolutional backbones with attention among patches for image classification. The same idea has been incorporated into network designs for semantic segmentation [212, 249, 35], object detection [25], and panoptic segmentation [35]. Transformers have been introduced recently also for video processing. TrackFormer [142] uses a transformer architecture for multi-object tracking. ViViT [6] and TimeSFormer [15] use ViT-like patches from multiple frames for video classification.

The main bottleneck of these approaches are the prohibitive memory requirements. TrackFormer [142] can only process 2 images at a time, which prevents end-to-end training on the whole video. Similarly, the all-to-all patch attention, which is the cornerstone of [15, 6], comes with often infeasible memory requirements. ViViT [6] needs the combined memory of 32 TPU accelerators to process a single batch of 128 frames. Our work overcomes these limitations by using sparse attention between 2D and 3D features. The sparsity is achieved by using image formation constraints directly from the poses of the cameras, which significantly reduces the memory requirements. For reference, RayTran processes up to 96 frames of a video and reconstructs all instances on a single 16 GB GPU.

3D using a dedicated depth sensor. Our work draws inspiration from several 3D object reconstruction methods that directly work on point clouds obtained by fusing RGB-D video frames. Early works use known pre-scanned objects [195], hand-crafted features [156, 62, 123, 206], and human intervention [206]. Recent works use deep networks to directly align shapes on the dense point clouds [8, 7, 9, 89, 205]. Fei et al. [60] align a known set of shapes on a video in 4 DoF, by using a camera with an inertial sensor.

Using an additional sensor reduces the search-space required to accurately re-construct an object in 3D. Both the depth and the inertial sensors eliminate the depth-scale ambiguity, and compared to re-constructing from pure RGB, RGB-D sensors provide cleaner, much more realistic results. Our work does not require the intermediate step of point-based reconstruction, does not use the extra depth sensor, and can directly reconstruct objects in a posed RGB video.

3D detection and reconstruction from a single image. Pioneering works in this area process a single image to either infer the pose of an object as an oriented 3D bounding box [152, 140], or to also predict the 3D shape of the object [227, 145, 36, 66, 231, 234, 163, 33]. Works that are able to predict an output for multiple object instances, typically first detect them in the 2D image, and then reconstruct their 3D pose and/or shape [84, 67, 111, 90, 220, 110, 159, 167, 112, 57, 82, 72].

3D predictions from single images tend to be inaccurate due to scale-depth ambiguity, and often methods of this category compensate for it in a variety of ways, e.g., based on estimating an approximate pixel-wise depth map from the input image [84], by requiring manually provided objects' depth and/or scale [67, 111, 112] at test time, or by estimating the position of a planar floor in the scene and assuming that all objects rest on it [90]. Some works [220, 159, 167] attempt to predict object depth and scale directly based on image appearance. Our proposed approach processes multiple frames simultaneously, and implicitly compensates for the scale-depth ambiguity by using many different view-points of the objects appearing in the scene.

3.4 Proposed Approach

Our method takes multiple views (video frames) of a scene and their camera parameters as input. Each view captures a different part of the same 3D scene. It outputs the 3D pose (rotation, translation, scale), the class, and the 3D shape of all objects in the scene.

We achieve this with a single, end-to-end trainable, neural network model. We propose a transformer-based backbone that processes the input views and infers a global 3D volume representation for the entire scene. We use this representation to predict the object shapes, poses, and classes, by attaching a DETR-style [25] head to it. In addition, we perform two additional auxiliary tasks: 3D occupancy, where we predict coarse binary volumetric occupancy for all objects together, and 2D foreground-background amodal segmentation. The overview of our architecture is illustrated in Fig. 3.1.

3.4.1 The RayTran Backbone

We propose a neural network architecture that operates on two alternative representations in parallel. The first one is three-dimensional and describes the 3D space that the scene occupies. We use a voxel grid V with *global* features that coincides with this space. The second one is two-dimensional and describes the scene from the perspective of the individual views. For each

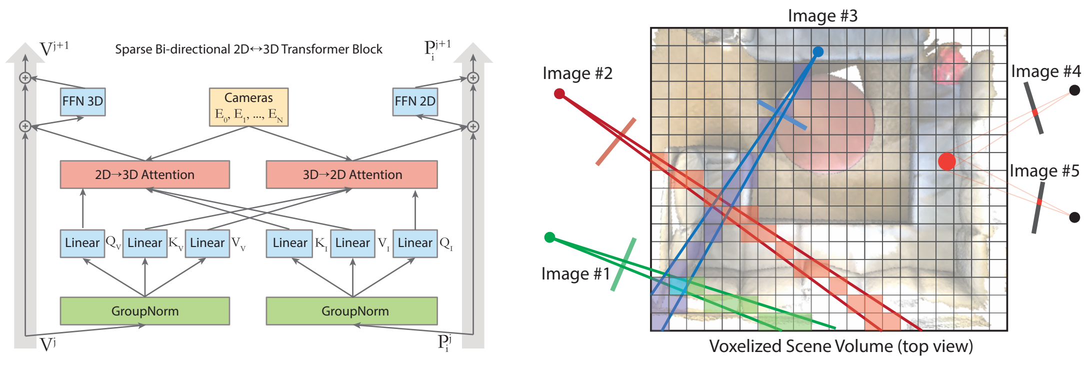


Figure 3.2: **$2D \Leftrightarrow 3D$ ray-traced transformer block (left).** Each block uses two parallel residual network streams that exchange information by attention. They consist of two layers of ray-traced sparse attention (2D→3D and 3D→2D) followed by a feed-forward network (FFN) composed of 3D and 2D convolutions, respectively. The voxel features (3D) inform the image features (2D) at each stage of the backbone. The 3D reconstruction head uses the voxel features (output of left stream), whereas the 2D foreground-background segmentation head uses the pixel grid (output of right stream).
Intertwining 3D voxel- with 2D image features (right): Multiple voxels can project on the same pixel, and multiple pixels from multiple cameras can look at the same voxel. The proposed attention layer models this interaction in an intuitive way.

view $i = 1..N$, we use a pixel grid P_i of *image* features that coincides with the view's image.

The two representations are connected implicitly through the image formation process. We model this as a sequence of $2D \Leftrightarrow 3D$ neural network transformer blocks (Figure 3.2, left). The j-th block takes all views $P^j_{1...N}$ and the volume V^j as input, mixes their features, and outputs a pair of new representations ($P^{j+1}_{1...N}$ and V^{j+1}). This allows the global 3D representations to be progressively populated by local features from the different views, while at same time the 2D representations progressively accumulate global features in different depths of the network.

The output of the RayTran backbone is a 3D feature representation of the scene, derived from the input views. In order to compute the initial 2D representation P^0_i, we embed ResNet-18 [76] in our backbone (pre-trained on ImageNet). We run ResNet-18 over the input views i and we take the output of its last block for each view. To initialize the 3D volume representation V^0, we cast a ray (un-project) from all the pixels P^0_i onto the 3D volume. We then average the image features that fall into each voxel of V^0.

Block operation. The 2D⇔3D blocks of RayTran consist of two parallel network streams, as shown in Figure 3.2 (left). The first one ($2D \Rightarrow 3D$), mixes features from P^j_i into V^j and outputs V^{j+1}. The second one ($3D \Rightarrow 2D$) from V^j into P^j_i, resulting in P^{j+1}_i. We propose to build both networks using the multi-headed attention mechanism [225].

The attention mechanism can translate an input vector (1D array of features) from a source domain into a differently-sized vector in a target domain. To do this, the mechanism computes a *key* vector that describes each position in the source domain and a *query* vector that describes each position in the target domain. It then computes a matrix describing the relation between source and target positions, by storing the dot product between the features at position i in the *key* and position j in the *query* at (i, j) in the matrix. Finally, the mechanism computes a *value* vector from the input vector and multiplies this with the attention matrix in order to obtain the output. The *key* and the *value* depend on the input vector (from the source domain), while the *query* depends on a vector from the target domain. The goal of the attention mechanism is to learn the dependencies between the two domains.

The attention mechanism is intrinsically well suited to model the connection between pixels and voxels. Multiple pixels from multiple cameras can look at the same voxel, as shown in Fig. 3.2 (right). We need a mechanism to consolidate their features in the voxel. Similarly, multiple voxels can project onto the same pixel and we need to consolidate their features. The matrix-*value* multiplication in the attention mechanism naturally achieves the desired effect.

For $2D \Rightarrow 3D$ attention, we derive the *key* and the *value* from all pixels from all views of P_i^j and the *query* from all voxels of V^j. For $3D \Rightarrow 2D$ attention, conversely, from V^j and P_i^j. We introduce skip connections in both networks, by adding the inputs of the attention mechanism to its outputs. We then post-process with a feed-forward network, built with 3D and 2D convolution layers respectively (Fig. 3.2).

Ray-traced attention layers. In a realistic setting, the attention layer has infeasible memory requirements. Our backbone operates on multiple frames simultaneously, 20 during training and 96 at inference time, each using a 2D feature grid of 40×30 for the 2D features P_i. We use a voxel grid with resolution $48 \times 48 \times 16$ to model a $9m \times 9m \times 3.5m$ volume, corresponding to voxel dimensions of approximately $19cm \times 19cm \times 22cm$. We use 256 features in both the 2D and 3D representations and 8 heads in the attention layers. Given the above numbers, the attention matrices in each 2D $\Leftrightarrow$ 3D block alone would require $\approx$ 52GB of memory with 20 frames, which is prohibitive.

To overcome this, we embed knowledge about the image formation process into the architecture (Fig. 3.3). A pixel and a voxel can interact with each other directly only if there is a camera ray that passes through both of them. If no such ray exists, the two are unlikely to interact, and we set the corresponding entry in the attention matrix to zero. This is mathematically equivalent to the masking mechanism employed in autoregressive transformers to enforce causality [225], but crucially allows us to store the matrix in sparse form and significantly reduce memory consumption. A pixel can only interact with $O(\sqrt[3]{|V|})$ voxels, where $|V|$ is the number of voxels in V, since any ray can only pass through at most this many voxels. We thus need $O(|V|^{\frac{2}{3}})$ times less memory to store the matrix. In sparse coordinate format, which encodes each matrix entry with 3 numbers (row, column, value), the matrix from our example above would consume 270 times less memory (3×64.4MB instead of 52GB). We call multi-headed attention based on such sparse matrices *ray-traced sparse attention*.

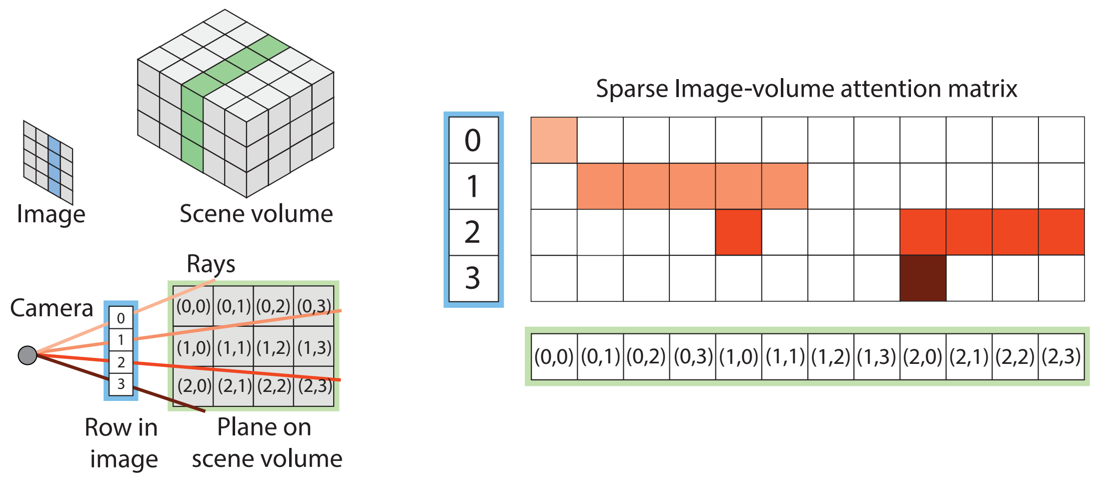


Figure 3.3: **Ray-traced sparse attention.** A pixel and a voxel are likely to interact only if a ray passes through both of them. We exploit this to significantly reduce the memory requirements of our 2D ⇔ 3D blocks, by sparsifying the attention matrices. If no ray passes though a pixel/voxel pair, the two are unlikely to interact and we omit the corresponding value in the matrix.

We use the camera parameters to determine which pixel-voxel pairs interact with each other. In turn, the camera parameters can be computed with off-the-shelf pipelines such as COLMAP [202]. To make full use of the limited volume that our backbone can focus on, we center the camera positions within it.

3.4.2 Task-specific heads on top of the backbone

3D pose estimation and shape reconstruction. For our main task, we predict the 3D pose, and reconstruct the shape of all objects seen in the video. We use a DETR-style [25] architecture, with multi-headed attention between 64 object query slots and the voxels in the backbone output. In each slot, we predict the object's class, its shape in canonical pose, 3D center, 3D anisotropic scale, and 3D rotation. We use a special *padding* class to indicate that a query slot does not contain a valid object. We encode the shape as a 63×63×63 voxel grid, which we predict with a sequence of transposed 3D convolution layers from the query's embedding. We use Marching Cubes [118] to convert the voxel grid to a mesh. We only predict rotation around the 'up'-axis of each object (as one angle), as most objects in our dataset are only rotated along this axis.

We use cross entropy for predicting the class, binary cross entropy for the shape's voxels, soft L_1 loss for the object center, L_1 loss over the logarithm of the scales, and a soft L_1 loss for the rotation angles. Finally, we match predictions to ground-truth objects in DETR using a linear combination of all losses except the voxel one, which we exclude for performance reasons. As in [25], we supervise at all intermediate layers.

Family	Method	**class avg.**	**global avg.**	bathtub	bookshelf	cabinet	chair	display	sofa	table	trashbin	other
Single-frame baselines	Mask2CAD [111] +avg	2.5	3.5	0.0	1.9	1.5	6.8	3.7	2.7	1.4	3.0	1.2
	Mask2CAD [111] +pred	11.6	16.0	8.3	3.8	5.4	30.9	17.3	5.3	7.1	25.9	0.5
Multi-Frame Methods	MVS [53] + RGB-D fitter [7]	18.8	21.7	15.8	8.5	17.3	34.3	25.7	15.0	10.9	35.8	6.1
	ODAM [121]	25.6	29.2	24.2	12.3	13.1.	42.8	36.6	28.3	31.1	42.2	0.0
	Vid2CAD [141]	30.7	38.6	28.3	12.3	23.8	64.6	37.7	26.5	28.9	47.8	6.6
	RayTran	**36.2**	**43.0**	19.2	34.4	36.2	59.3	30.4	44.2	42.5	31.5	27.8

Table 3.1: Quantitative results on the Scan2CAD [8] dataset using the original Scan2CAD metrics. Results for Mask2CAD variants, MVS+RGB-D fitter, and Vid2CAD are as reported in [141]. ODAM originally reports in another metric (Tab. 3.2). We re-evaluate in the Scan2CAD metrics based on model outputs provided to us by the authors. Note that ODAM was not trained to predict the 'other' class. When excluding it from the metrics, ODAM achieves class avg. of 28.8% and global avg. of 33.5%.

3D occupancy prediction. As an auxiliary task, we also predict the binary 3D occupancy of all objects for the whole scene on a coarse voxel grid. We use one 3D convolution layer on top of the backbone output, and we supervise with binary-cross-entropy. We run occupancy prediction as an auxiliary task, to directly teach the network about the combined object geometry. This is crucial for the 3D object reconstruction task, as the DETR-style head fails to pick up any training signal if the network is trained without it.

2D foreground-background (FG/BG) segmentation. As a second auxiliary task, we predict a 2D *amodal* segmentation mask in each view, for all objects together. A pixel belongs to the mask if it lies on any object in the view, regardless of occlusion. We use transposed convolutions, combined with non-linearities and normalization layers, to up-sample the pixel stream output P_i^n of the last 2D ⇔ 3D block to the original input resolution (16-fold). We supervise using the binary cross entropy loss. We create the amodal masks by rasterizing the combined geometry of all ground-truth 3D objects into each view. Predicting amodal masks enhances the backbone's 3D understanding of the world. In general, the amodal mask is ill-defined for occluded regions in a single image. It becomes well defined with multiple views however, if some of them observe the object behind the occluder. Hence, this FG/BG task pushes our network to reason about geometric relations across multiple views.

Novel View Synthesis. While we focus on multi-object 3D reconstruction, our backbone and the scene-level representation it outputs can be used for other 3D tasks as well. In the appendix, we provide qualitative results for Novel View Synthesis, which builds upon RayTran's backbone.

3.5 Experiments

Datasets and evaluation metrics. We evaluate our method on Scan2CAD [8], following their protocol and evaluation metrics. Concretely, we use videos from ScanNet [42], 3D CAD models

Prec./Rec./F1	MOLTR [120]	ODAM [121]	Vid2CAD [141]	ImVoxelNet [43]	RayTran
@IoU> 0.25	54.2/55.8/55.0	64.7/58.6/61.5	56.9/55.7/56.3	52.9/53.2/53.0	**65.4/61.8/63.6**
@IoU> 0.5	15.2/17.1/16.0	31.2/28.3/29.7	34.2/33.5/33.9	17.0/17.1/17.0	**41.9/39.6/40.7**

Table 3.2: Quantitative results on Scan2CAD using the ODAM [121] metrics. To evaluate RayTran, we derive an oriented 3D box by using the 3D transformations predicted by the model. RayTran outperforms all other works, especially at the stricter IoU threshold (@IoU> 0.5), showing it produces particularly accurate object poses. Note that for Vid2CAD we report the updated results from https://github.com/likojack/ODAM (which match exactly the Vid2CAD paper [141]). Also note that ImVoxelNet outputs axis-aligned boxes, which hinders its performance at high IoU thresholds. Finally, results for MOLTR and ODAM are as reported in [121].

from ShapeNetCore [26], and annotations that connect the two from Scan2CAD [8]. ScanNet provides videos of rich indoor scenes with multiple objects in complex spatial arrangements. ShapeNetCore provides CAD models from 55 object classes, in a canonical orientation within a class. Scan2CAD provides manual 9-DoF alignments of ShapeNetCore models onto ScanNet scenes for 9 super-classes.

We use these datasets both for training and evaluation. During training, we consider all ScanNet videos in the official train split whose scenes have Scan2CAD annotations (1194 videos). We evaluate on the 306 videos of the validation set, containing a total of 3184 aligned 3D objects. We quantify performance using the original Scan2CAD metrics [8] and the metrics introduced in ODAM [121]. In the Scan2CAD metrics, a ground-truth 3D object is considered accurately detected if one of the objects output by the model matches its class and pose alignment (passing three error thresholds at the same time: 20% scale, $20°$ rotation, 20cm translation). We report accuracy averaged over classes ('class avg.') as well as over all object instances ('global avg'). In the metrics of [121], an object is considered accurately detected if the Intersection-over-Union (IoU) of its oriented 3D bounding box to a ground-truth box of the same class is above a predefined threshold. We report precision, recall, and F1 score. Finally, the dataset also provides dense 3D meshes for the scene produced using a dedicated depth sensor. We ignore this data, both at training and test time (in contrast to some previous works which rely on it [8, 7, 9, 89, 205]).

Training details. We implement our model in PyTorch [164]. We train on 20 frames per video, using 16-bit float arithmetic. This allows us to fit one video on a GPU with 16GB of memory. We use 8 GPUs in total, resulting in a batch size of 8. We train RayTran in three stages. We first train just the backbone for 224k steps (1500 epochs) on the task of predicting 3D occupancy (Sec. 3.4.2). We then enable all other tasks except the shape predictor and we train for another 239k steps (1604 epochs). Finally, we train just the shape predictor for another 5k steps (17 epochs), after freezing the rest of the network parameters. We use the AdamW [132] optimizer, with a learning rate of 10^{-4} and weight decay $5 \cdot 10^{-2}$.

Compared methods. We compare RayTran against Vid2CAD [141], ODAM [121], MOLTR [120], and ImVoxelNet [43], four recent methods for 3D object pose estimation and detection from RGB videos.

We further compare to two baselines that process frames individually, defined by [141]. These extend Mask2CAD [111], which in its original form does not predict the 3D depth nor the scale of the object. The first baseline, 'Mask2CAD +avg', estimates an object's depth and scale by taking the average over its class instances in the training set. The second baseline, 'Mask2CAD +pred', predicts the scale of the actual object in the image (and then derives its depth from it). Both baselines aggregate 3D object predictions across all video frames and remove duplicates that occupy the same volume in 3D.

Several previous methods report strong results on Scan2CAD by using a dedicated RGB-D depth sensor to acquire a dense 3D point-cloud of the scene. Those methods have an intrinsic advantage and operate by directly fitting CAD models on the scene's 3D point cloud [8, 7, 9, 89]. Instead, our method only uses the RGB frames. Hence, we compare to a strong alternative method, defined in [141], that replaces the input of the best RGB-D fitting method [7] with 3D point-clouds generated by the state-of-the-art multi-view stereo method DVMVS [53]. We train DVMVS on ScanNet, and re-train [7] on its output.

Main results. Tab. 3.1 shows the results in the Scan2CAD metrics. RayTran outperforms both single-frame baselines as well as the 'MVS + RGBD fitter' combination by a wide margin (+33.7%, +24.6%, +17.4% class avg. accuracy respectively). RayTran also outperforms both competitors that align CAD model to RGB videos but rely on tracking: Vid2CAD [141] (+5.5%) and ODAM [121] (+10.6%). Importantly, RayTran is also much simper in design, as [141, 121] consist of multiple disjoint steps (object detection, tracking, multi-view optimization). Fig. 3.5 and Fig. 3.6 illustrate qualitative results for our method.

Looking at individual categories, we obtain the best result on 5 out of 9, and in particular on the "other" category, which is hard for methods based on retrieving CAD models [141, 8, 7]. Our method instead predicts 3D shapes as voxel grids which helps to generalize better and to adapt to the large variety of object shapes in this catch-all category. On 'trashbin' we do moderately worse, possibly because of the relatively coarse voxel resolution of the backbone representation.

For completeness, we also compare to methods [8, 7] in their original form, i.e. fitting CAD models to high-quality dense RGB-D scans. Surprisingly, RayTran (36.2%/43.0%) improves over [8] (35.6%/31.7%), despite using only RGB video as input. While the state-of-the-art [7] performs even better (44.6%/50.7%), this family of methods are limited to videos acquired by RGB-D sensors.

Fig. 3.4 reports accuracy for each transformation type separately (translation, rotation, and scales). Our method predicts all transformation types better than Vid2CAD, ODAM, and the best single-frame baseline Mask2CAD+pred. As objects are considered accurately detected only

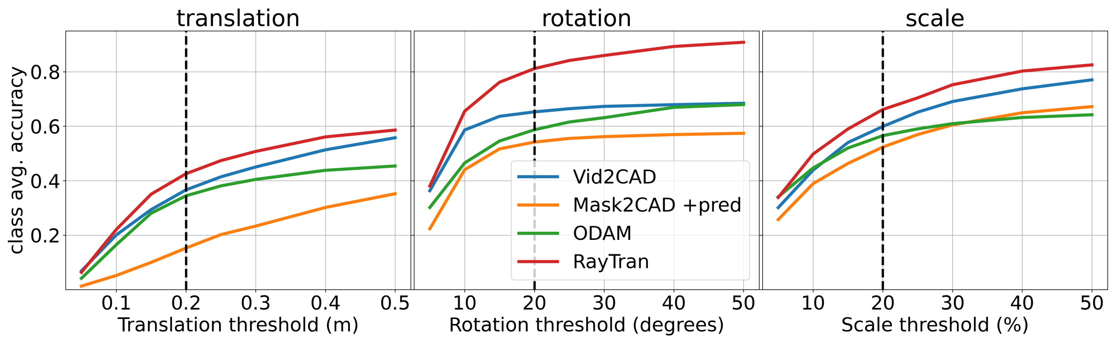


Figure 3.4: **Transformation type ablation**: Class-avg accuracy as a function of the evaluation threshold (the vertical dotted line shows the default value, used in Tab. 3.1). We examine each transformation type separately. RayTran achieves better accuracy than Vid2CAD, ODAM, and 'Mask2CAD +pred' on all transformation types.

Method.	extra input	auxiliary tasks	class avg.	global avg.
RayTran	-	3D occupancy + 2D FG/BG seg.	36.2	43.0
RayTran w/o FG/BG.	-	3D occupancy	33.8	40.1
RayTran + GT masks	2D GT masks	3D occupancy + 2D FG/BG seg.	47.6	52.5

Table 3.3: Effects of object segmentation in RayTran. Without FG/BG segmentation as an auxiliary task, the network performs worse (first two rows). If we grant the model the ground-truth segmentation masks as input, results substantially improve, highlighting how future progress on automatic 2D segmentation will benefit our work too (last row). In all cases, the model is trained with the main 3D object pose/shape estimation loss (Sec. 3.4.2), in addition to the auxiliary losses listed here.

when passing all 3 thresholds *simultaneously*, improving translation is the biggest avenue for improving our overall quantitative results (Tab. 3.1).

Tab. 3.2 reports results in the ODAM metrics, which allow us to compare to MOLTR [120] and ImVoxelNet [43]. We choose an object score threshold to maximize the F1 score on the val set for methods that predict object scores (RayTran, Vid2CAD, ImVoxelNet), following the practice of [121]. RayTran outperforms all four methods [121, 120, 43, 141] at both IoU thresholds. ImVoxelNet [43] reports results on ScanNet, not on Scan2CAD. To compare properly we use their publicly available source code and re-train on Scan2CAD. The original code only outputs axis-aligned object boxes on ScanNet (and hence on Scan2CAD, which is derived from it). This prevents comparison on the Scan2CAD metrics, as we cannot compute precise rotation and scale components. Finally, predicting box rotation could potentially improve the results in the ODAM metrics.

Ablation: 2D FG/BG segmentation as auxiliary task. Our model predicts amodal masks, which reinforces the backbone's 3D understanding. Pixels where the object is occluded can only be predicted correctly in 2D as part of the amodal mask by relying on signal from other frames, via the global 3D representation. To support this claim, we trained a version of our model where we disabled the 2D FG/BG segmentation auxiliary task of Sec. 3.4.2. This reduces class-avg accuracy by -2.4% (36.2% vs. 33.8 first two rows of Tab. 3.3).

Ablation: perfect segmentation. Our model performs both 2D and 3D analysis. The main challenge in the 2D analysis is pixelwise segmentation in the input frames. We explore here what would happen if our model were granted perfect object segmentation as input. We train a model which inputs a binary mask as a 4th channel, in addition to RGB. A pixel in the mask is on if it belongs to any object of the 9 classes annotated in Scan2CAD, and 0 otherwise. This augmented model improves class-avg accuracy by +11.4% (reaching 47.6%), and global-avg by +9.5% (reaching 52.5%, Tab. 3.3 last row). Hence, as research on 2D segmentation improves, so will our model's 3D scene understanding ability.

Ablation: number of objects in the scene. By design, our network cannot predict more object instances than the query slots in the DETR head (64). Moreover, typically only about 30% of all query slots bind to an actual object [25] in scenes containing many objects. This limits recall on such scenes, which sometimes do occur in Scan2CAD. Carion et al [25] believe that query slots tend to bind to fixed spatial regions, regardless of the content of a test image, causing this limitation. We operate in 3D, which likely exacerbates it because we need many more queries to cover the 3D space.

To understand the effect of this phenomenon on our model's performance, we evaluate here on 3 subsets of Scan2CAD's val split, containing scenes with *at most* 10, 20, and 30 objects respectively. This reduces the number of objects undetected by the fixed 64 query slots in our DETR head.

The class-avg accuracy of RayTran indeed improves in scenes containing fewer objects (from 36.2% in all scenes, up to 37.4% in scenes with < 10 objects). The accuracy of the best previous method Vid2CAD instead remains constant. In scenes with at most 10 objects, we outperform Vid2CAD by 6.7% (37.4% vs. 30.7%), which is a larger difference than on all scenes (5.5%: 36.2% vs. 30.7%). Hence, DETR's limitation is affecting our model as well and improving upon it will improve our overall performance.

Ablation: number of input frames. Our model can process a variable number of input frames per video. We use 20 frames at training time to limit memory requirements. In all experiments so far, we used 96 frames at inference time, as using more frames improves coverage of the 3D volume of the scene and hence accuracy of the output. To support this claim, we now reduce the number of frames at inference time. With 48 frames class-avg. falls by -0.4%. Worse yet, if inference were constrained to 20 frames as during training, then performance would drop by -3.3%. This highlights the value of our model's ability to input a variable number of frames.

3.6 Conclusions

We presented RayTran, a novel backbone architecture for 3D scene reconstruction from RGB video frames, that uses transformers for unprojecting 2D features and consolidating them into a global 3D representation. We introduced the ray-traced sparse transformer block, which enables feature sharing between the 2D and 3D network streams, in a computationally feasible way on current hardware. We use this architecture to perform 3D object reconstruction for the full scene by combining it with a DETR-style network head. Our architecture can reconstruct the whole scene in a single pass, is end-to-end trainable, and does not rely on tracking. We perform experiments on the Scan2CAD benchmark, where RayTran outperforms (1) recent state-of-the-art methods [141, 121, 120, 43] for 3D object pose estimation from RGB videos; and (2) a strong alternative method combining Multi-view Stereo [53] with RGB-D CAD alignment [7].

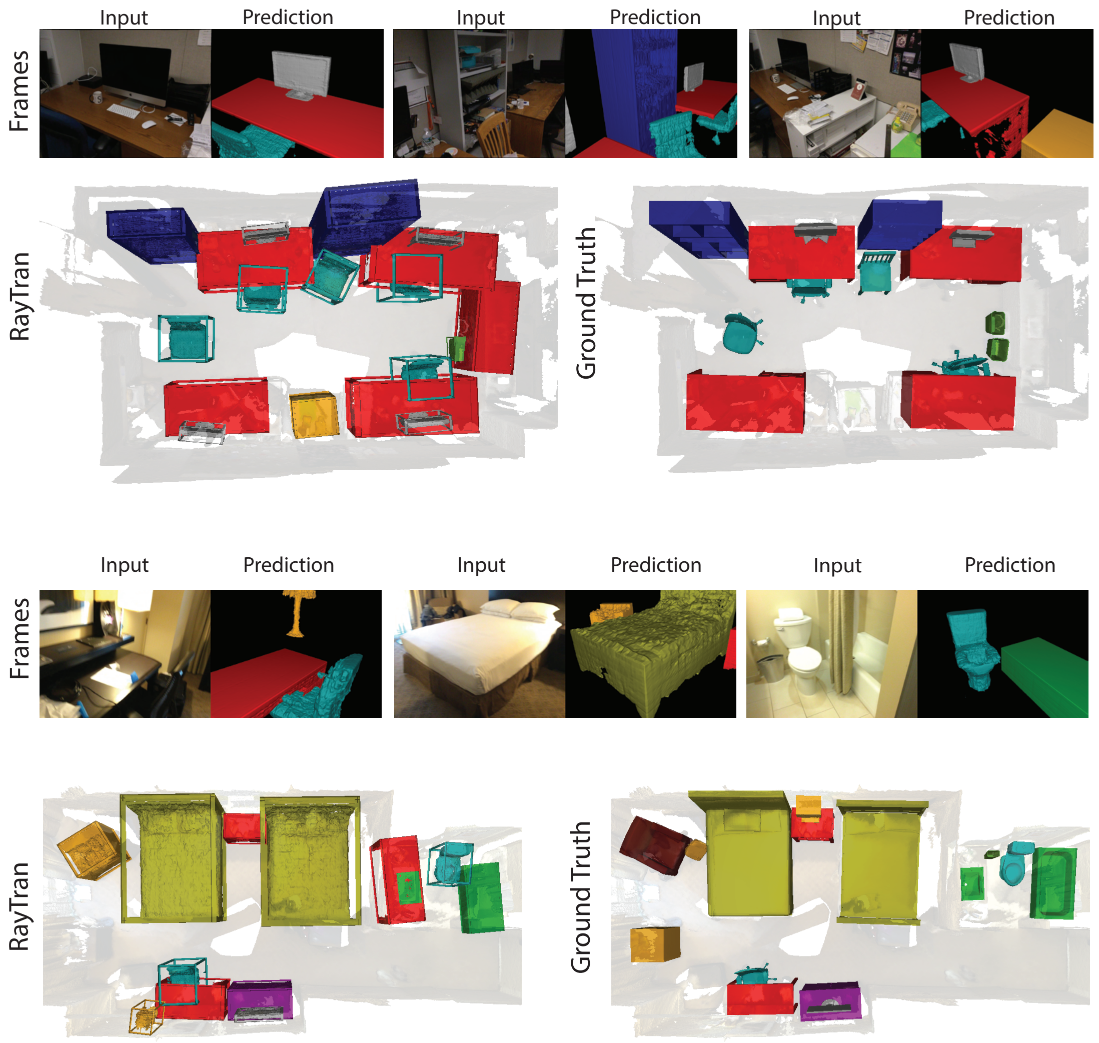


Figure 3.5: **Qualitative Results (top-view, with frame overlays)**: We show the 3D pose estimation (as oriented boxes) and shape reconstruction outputs of RayTran against the ground truth, from the top and from the viewpoint of the images. The objects are colored by class. We are able to reconstruct complex scenes in a single pass.

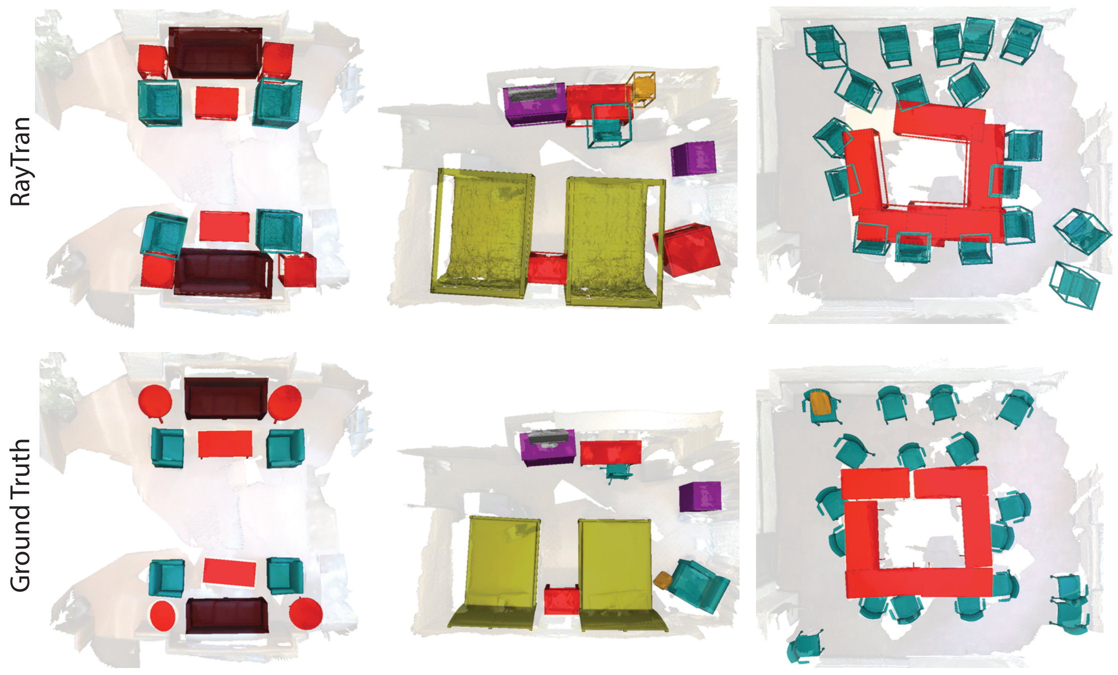


Figure 3.6: **Additional qualitative Results (top-view)**: We show the 3D pose estimation and shape reconstruction outputs for 3 additional scenes. For each detected object we visualize its 3D oriented bounding box, as well as its reconstructed mesh.

3.7 Project background

RayTran is the result of a 2021 summer internship at Google Zurich (remote) with the goal of detection and 9DOF posing of household objects from videos of the Vid2CAD [141] dataset. The initial ideas, discussed before the internship start, revolved around incrementally building on prior work. While we finally settled on a more bold design, in this section we briefly discuss these directions to show the alternative considerations. We argue that the final design combines many of their benefits while avoiding the downsides.

Local features matching. One proposal was to reuse the ideas in DISK [221] and learn a multimodal keypoint detector and descriptor, sharing the embedding space between video frames and a canonical representation of object prototypes (CAD models) stored in a database. In practice, the latter can be 3D keypoints on the CAD models themselves or 2D keypoints on a set of canonical renders. In either case, the model would be trained with the domain gap in mind, resulting in a 2D-3D template matching algorithm. Locality is a double edged sword and this approach inherits both the strengths and the weaknesses of local features. On one hand, their modularity allows for post-training updates to the prototype database and makes the method robust to partial occlusions. On the other, local approaches ignore context *by design*, making treatment of partially similar objects difficult; post-hoc attempts to use context usually lead to system complexity and suboptimal performance. Aside from the tradeoff highlighted above, there are further problems to address. Given a number of candidate local matches from a video frame to a database of CAD prototypes it is unclear how to resolve the best CAD match given many candidates and cluster individual objects, especially with multiple instances of the same shape.

Multiple object tracking. The other original proposal was to extend Mask2CAD [111], a single-frame 3D-aware object detector and pose estimator, with the capacity to leverage information from multiple video frames. A natural paradigm is *tracking by detection*, that is to detect objects in all frames individually and merge them into tracklets based on appearance similarity and motion prediction. Compared to the usual tracking setting with a fixed camera and moving objects, in Vid2CAD this is flipped with static objects and a moving camera. With precise camera motion estimate, obtained from SfM, the main difficulty lies in a) canonicalizing detected objects' poses while accounting for uncertainty specific to each frame and b) gradual unification of detections in global coordinate frame. Importantly, the scale-depth ambiguity means that the translation uncertainty along the camera viewing axis (z) is much larger than in the camera plane ((x, y)), with the axes x, y, z being different for each video frame. A variant, proposed by Stefan Popov, was to use a 3D voxel grid as a representation of the entire scene, unproject all view features into it and then project them back to views. With the detection still happening on a per-view basis, this method provides global appearance cues to Mask2CAD, helping with occlusions and scale ambiguities. A problem with this method is the lack of depth information making the unprojection itself ambiguous.

Representing appearance. Both the multiple object tracking and the local feature-based designs assumed representing object appearances with CAD models. A database of artist-designed 3D models, provided with the training dataset, would be retrieved from at inference time. The retrieved model is then emplaced in the 3D scene using the predicted 9DOF parameters. This approach ensures aesthetically pleasing results but leads to problems in quantifying errors and in loss design. CAD model retrieval can be formulated as multiclass classification among the N different shapes but this does not reflect that retrieving a wrong but similar model should be less problematic than a different object category altogether. This is especially important as the CAD model database (ShapeNet [26]) was created independently from the scan dataset and the object-model annotations are on best effort basis, rarely perfectly matching the true shape. At the planning stage we assumed an approach of learning a continous shape embedding via an autoencoder and retrieving the nearest neighbor in its latent space. This would bring in a notion of shape similarity, making errors more quantifiable.

Transformers in vision. The discussion on the scope of the internship took place while transformers [225] rapidly expanded past natural language processing. DETR [25] was posted to arXiv 14 months before the internship start, ViT [51] 9 months, TimeSFormer [15] less than 5 and ViViT [6] 4. Transformers were the first to reliably abstract away the raster layout of images, greatly simplifying architectures where the inductive bias of CNNs is not beneficial.

One such case are video transformers (ViViT and TimeSFormer), where video understanding benefits from the possibly non-local attention patterns, which is helpful when entities move arbitrarily from frame to frame. Addressing the same problem with CNNs requires drastically growing the receptive field. It was clear that such a video transformer architecture, combined with per-token encoding of camera pose, would have the power to connect the patches belonging to individual objects in the scene across time and create a global representation.

The representation can then be queried with a DETR-like detector to obtain useful object detections. DETR is another instance of benefit from ditching the raster layout of images, with a transformer detecting individual objects while attending to arbitrary image patches, as their shapes dictate. This idea naturally extends to tokens representing video streams across time.

The problem is however with the scale of this system, in two ways. Firstly, even with the space-time factorization in video transformers, the memory cost of such an architecture is very large and although not infeasible, it exceeded the project's budget. Hardware-aware implementations like Flash Attention [44] change this equation significantly but were not available at the time of writing. Secondly, such a powerful network with very limited inductive biases requires a very strong supervision signal, which is not available for the task. Despite the large number of video *frames* in ScanNet [42], from which Vid2CAD is derived, the number of video *sequences*, and therefore unique room layouts, is limited, giving raise to issues like overfitting to specific object configurations.

RayTran. The final concept was born in the first two weeks of the internship and combines the use of transformers, championed by the candidate, with Stefan Popov's idea of a voxel scene representation. It leverages the strength of transformers while addressing the two issues highlighted above. By constraining the information flow in a geometrically motivated way, it imposes an inductive bias, compared a standard video transformer where each patch can interact with any other. This greatly reduces the expressivity of the model without impacting the connections which are useful for the task. This sparsity of attention is also responsible for mitigating the computational cost of the algorithm, making the model easily trainable on 8×16GB GPUs. At the same time, thanks to a unified scene representation, RayTran avoids the notorious issues of multiple object tracking caused by tracklet segmentation — especially on objects coming in and out of camera's view and on ones which are visually indistinguishable, such as industrially made furniture. Finally, RayTran departed from the initial concept by doing away with CAD model retrieval and instead predicting medium resolution voxelized shapes. This implicitly treats the Hamming distance between occupancy volumes as a shape similarity measure, leading to a simpler loss formulation and reasonable visual quality.

3.8 Future work

RayTran suffers from three main problems: the inefficiency of its spatial representation, the complexity of the supervised objective and the ensuing scarcity of suitable data. In the following subsection we expand on these points and suggest directions for incremental improvements. We then present a more revolutionary direction which leverages unsupervised learning to avoid the problem of data scarcity and discuss how such a system can learn from novel view synthesis. We expand on the problem of scene dynamics, which arises in this context as a nuissance factor precluding use of simple loss functions.

3.8.1 Limitations of RayTran

Inefficient spatial representation. The main problem with RayTran is the large memory footprint of the voxel scene representation. Although voxels have the advantage of easily interpretable geometry, making the sparse attention mechanism straightforward to implement, they lack the ability to adapt to the scene. Inevitably small, sub-voxel sized objects will be imprecisely localized while many voxels will be spent on representing empty space. The voxel grids account for the vast majority of RayTran's memory usage and are the reason why a node of 8×16GB GPUs was necessary to train the network. Developing a more adaptive scene representation, capable of focusing only on objects of interest while retaining precise geometrical information, is one of the biggest improvements to be made. Factorized NeRFs representation such as TensoRF [27] can be one source of inspiration.

Supervision complexity and data scarcity. Perhaps an even bigger issue, shared by other works in the space, is the complexity of task formulation, which involves separate labels and losses across rotation, translation, scale, classification and shape reconstruction. This results in a time-consuming annotation process, engineering effort spent on setting up data pipelines and in complex loss functions with many hyperparameters which require extensive tuning and make-or-break models. Overall, these factors harm reproducibility and hackability, greatly increasing the barrier to entry. This is in stark contrast to recent successes of multimodal models like CLIP [175] which show that simplifying the task formulation, even at the cost of increasing its difficulty, can lead to great results by shifting that burden from the limited researcher to the scalable model.

While we cannot propose a similar solution for 3D object detection, a good first step would be to move away from training directly on the costly 3D labels. With a weakly- or self-supervised objective for learning 3D perception it should be possible to reserve the task-specific annotations for converting the model's latent representations to a human-friendly format. This approach could also address the notorious problem with data scarcity, enabling use of larger but less annotated datasets, for example video frames with poses. Even though the Scan2CAD dataset contains 2.5M individual *images*, the number of *scenes* is only 1506, making overfitting a serious problem. Since the number of individual object layouts is equal to the number of scenes, a model quickly starts overfitting to specific rooms in the training set. This problem is acute enough that we conjecture that an efficient data augmentation scheme, however difficult it may be to formulate for this task, is likely to surpass the impact of many modelling innovations.

3.8.2 3D-GPT

Following our remarks on the task formulation and spatial representations, we provide a sketch of a system combining these properties. Inspired by the success of language models like GPT-3 [22] and BERT [47] at building rich and useful internal representations from masked token prediction, we propose to extend this idea to 3D vision by treating novel view synthesis as a pretraining task. This idea is not novel with recent prior work like the Scene Representation Transformer [194], RePAST [191], Object Scene Representation Transformer [193] and DORSal [91]. This body of work is largely driven by a specific group at Google Research and we argue that it shows the way to a more powerful and general successor to RayTran. Still, there are differences between our goals and the direction of those papers.

Data dimensionality and uncertainty. The big difference between modelling text and images is in the data dimensionality. With tokenization, the 8-128k attention window of transformers with efficient attention implementations [44] can hold useful amounts of text. At the same size, this is not enough to hold the pixels of a single RGB image at 256×256 resolution. Pixel-level autoregressive modelling is prohibitive and the models need to operate at a larger granularity. This is usually done in terms of image patches, which brings the problem of modelling intra-patch correlations. The most common approach ([194, 78]) is to ignore them altogether and

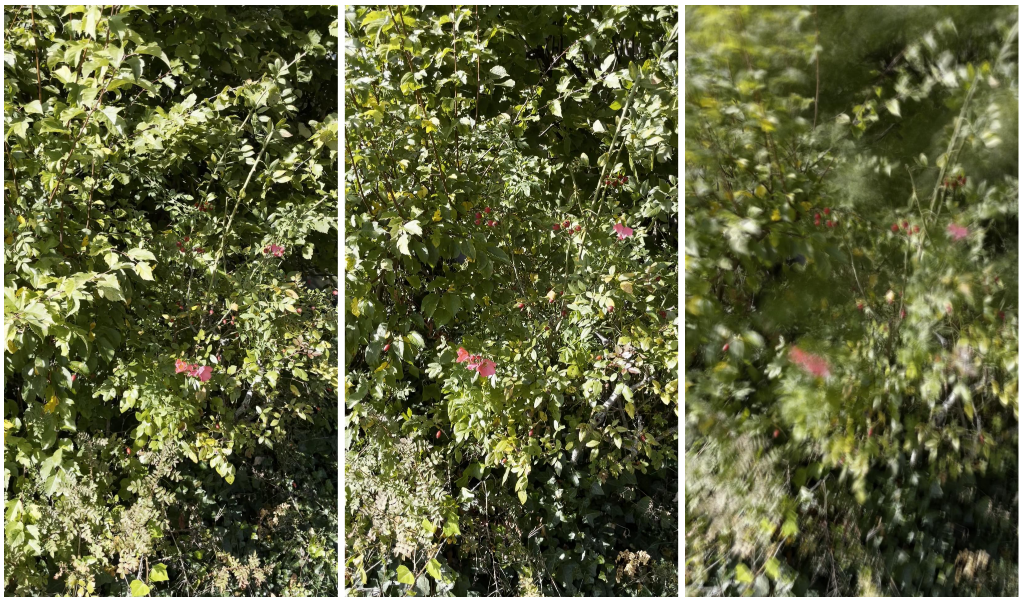


Figure 3.7: Foliage on a windy day. **Left**: still from POV A, **center**: still from POV B, **right**: time-average from POV B. A model seeking to maximize the likelihood of *sampling* the center image is incentivised to learn about leaf shapes and plant species. A model which regresses *the mean* (right image) will only look at rough color distribution. Best viewed zoomed.

use an L_2 loss, predicting the *mean* patch color. Faced with multimodal uncertainty, this means blurry results due to loss of high frequencies. This is in contrast to language models which autoregressively sample variables one at a time, modelling a richer set of correlations[1]. We argue that this deficit, aside from the visually unpleasant blurriness, leads to crucial loss of detail in network's internal representations. An example is given in Fig. 3.7. Despite easily recognizable multiple bush varieties in one view, the uncertainty caused by motion renders them and other visual features useless for regressing a neighboring view in the L_2 sense. This applies not only to the aleatory uncertainty of random wind gusts but also epistemic uncertainty of observing the top of a table and predicting its legs. The pixel-level appearance of the legs is dominated by their precise location, which can only be predicted if we already know the specific piece of furniture. Without this information, an L_2 novel view predictor will ignore cues such as the material and texture even though they are informative from a higher level perspective.

Sketch of the system. We argue that this lack of treatment of uncertainty is why the mentioned works [194, 191, 193, 91] fail to deliver generalizable 3D perception backbones[2]. We also posit that the emergence of diffusion generative models, discussed further in Sec. 4, shows a direction for addressing this issue in 3D vision models. We envision a model which uses the multiple input

[1]Autoregressive models of quantized patch embeddings like [179] are one workaround but they have other limitations.

[2][91] is very close to our vision, but they use a *frozen* perception backbone. This is likely due to difficulties with end-to-end training.

views to create a unified scene representation, like RayTran, and then uses this representation for *conditional* generation of novel views. More specifically, we propose to use two cross-attention based transformer modules and a generative 2D conditional diffusion model:

1. The first transformer, called *scene encoder*, attends to patches of input views, augmented with viewing geometry information (camera ray embedding), and deposits the extracted information in the *scene representation*, made of a fixed-size set of tokens with learnable initialization.
2. The second transformer, called the *query view decoder*, attends to the scene representation and transfers the relevant information to *query view tokens* which represent patches of novel views and initially contain only an encoding of camera rays.
3. Finally, the 2D diffusion model uses the query view tokens as conditional input to denoise (synthesize) a novel view.

Since this **transformer** model **learns 3D** perception for **downstream** tasks by **generating** novel views, it can indeed be called a 3D Generatively Pretrained Transformer: 3D-GPT. We acknowledge that this formulation reduces the denoising diffusion model to a variant of masked modelling objective which may become obsolete with the emergence of better paradigms for uncertainty-aware self-supervision. We propose the DDM model as known and tested technique with the additional benefit of easy visual inspection and debugging, but it is not central to our idea.

Downstream use. Our three-stage formulation makes the encoder not aware of the query views, forcing it to imbue the scene representation with information which may be relevant for *all* possible queries, ensuring rich latent representations for downstream tasks. The two remaining modules will not be used in most cases, as is often the case with self-supervised backbones. The query view decoder is responsible for querying the geometric information encoded in the latent representation. This is less interpretable than an explicit voxel-based lookup in RayTran but it instead lets the model optimize the latent structure to encode only the occupied areas, avoiding the inefficiency of voxels. Such "geometry-free" approaches have already been shown to work in a similar setting in [71] where depth prediction is learned while using novel view synthesis as a beneficial auxiliary task. Finally, the 2D diffusion model serves a double role. As a network, it is a "parametric loss function", similar to a decoder in autoencoder models, allowing strong compression in the information bottleneck. As a conditional generative model, it forces the query view tokens contain different granularities of data — from low frequency information useful for the general view layout to fine semantic detail such as plant species, useful for generating individual leaves. Such a system is capable of learning rich 3D representations from data which is cheap to collect at scale, such as Google Street View, and can be used for many downstream tasks. As an example, the DETR [25] detection head of RayTran is not voxel-specific and can be trained to ingest the tokenized scene representation of 3D-GPT instead. This will greatly reduce the risk of overfitting due to 3D-GPTs large pretraining corpus and strong (self-)supervision. We believe that over time such generalist models will replace task-specific algorithms like RayTran.

4 GECCO

“Gecco: Geometrically-conditioned point diffusion models” by Michał Tyszkiewicz, Pascal Fua, Eduard Trulls in ICCV2023. Candidate’s contributions: system design, implementation and evaluation.

4.1 Abstract

Diffusion models generating images conditionally on text, such as Dall-E 2 [177] and Stable Diffusion [187], have recently made a splash far beyond the computer vision community. Here, we tackle the related problem of generating point clouds, both unconditionally, and conditionally with images. For the latter, we introduce a novel geometrically-motivated conditioning scheme based on projecting sparse image features into the point cloud and attaching them to each individual point, at every step in the denoising process. This approach improves geometric consistency and yields greater fidelity than current methods relying on unstructured, global latent codes. Additionally, we show how to apply recent continuous-time diffusion schemes [211, 98]. Our method performs on par or above the state of art on conditional and unconditional experiments on synthetic data, while being faster, lighter, and delivering tractable likelihoods. We show it can also scale to diverse indoors scenes.

4.2 Introduction

Given the popularity of depth sensors and laser scanners, point clouds have become ubiquitous, with applications to robotics, autonomous driving, and augmented reality. Furthermore, they do not suffer from the precision/complexity trade-off inherent to voxel grids, and scale better and more generally than graph-based representations such as meshes. As a result, they have been extensively used for analytical tasks such as classification [171, 172, 214, 247, 173] and segmentation [171, 172, 83, 214, 65, 247, 251]. Recent work has turned to point cloud synthesis and its many applications to 3D content creation. However, this remains an emerging field and state of the art methods [236, 24, 103, 135, 250, 244] operate on small datasets that feature only a handful of object types [26]. More importantly, the generated shapes are typically not anchored in any prior and are thus difficult to control.

We present a novel approach that can mitigate these issues, taking our inspiration from generative

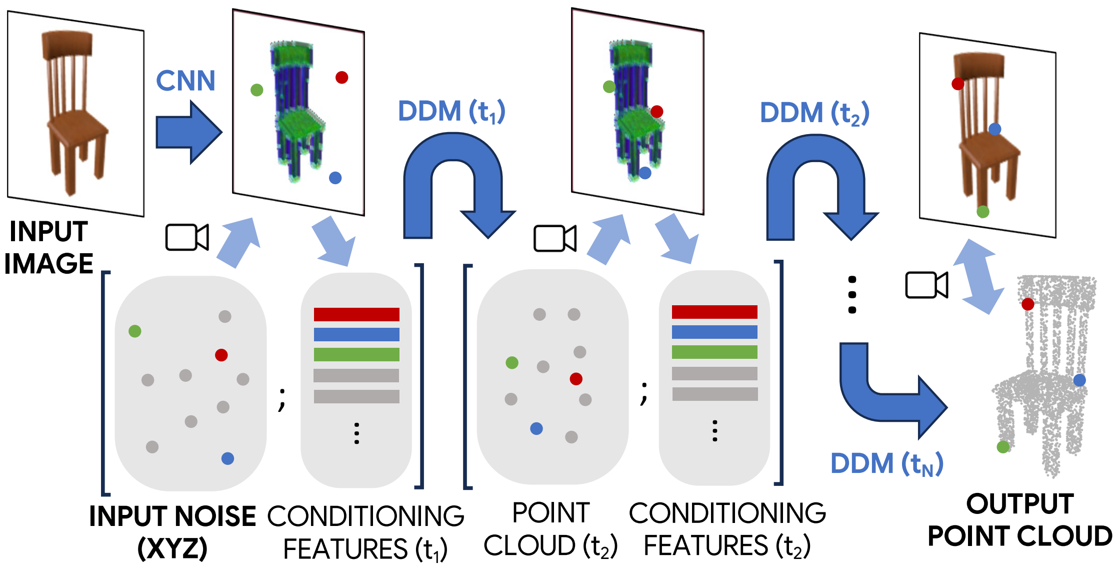


Figure 4.1: Our generative approach is based on denoising diffusion models (DDMs) and can be conditioned on images. At each denoising step we project the point cloud to the image, sample sparse features, and concatenate them to the locations, thus guiding the denoising process and yielding point clouds consistent with the images.

methods that perturb samples with a diffusion process [210, 98] and denoise them with a deep network, which can later be used to synthesize new samples by iteratively denoising a signal. Specifically, most denoising-based approaches generate novel samples from pure noise. But to mirror the success of text-based image synthesis [177, 187, 192], a generative approach must be able to not only produce samples of sufficient quality and diversity, but also ground them in contextual information. Applying this generic idea to point clouds is, however, not straightforward. We show how to achieve this by conditioning the network with sparse image features.

Unlike previous works relying on unstructured, global embeddings, we do so in a geometrically-principled way, by projecting the point cloud into an image, sampling sparse features at those locations, and feeding them to the network along with the point location, at each denoising step, as illustrated in Fig. 4.1. This allows us to render 3D objects geometrically and semantically consistent with the image content, while controlling the viewpoint. Unlike regression models, such as monocular depth, our method can generate plausible hypotheses for occluded regions. This work is thus a first step towards unlocking the applicability of denoising diffusion models to practical scenarios such as 3D content creation, generating priors for automotive or robotics applications, and single-view 3D reconstruction.

In short, we propose a novel generative point cloud model and show how to condition it on images. Our main contributions are:

1. We propose a framework composed of a permutation-equivariant Set Transformer [116]

trained with a continuous-time diffusion scheme, which performs on par with the state of the art on *unconditional synthesis* while running 10x faster and delivering exact probabilities.

2. We augment it with *geometrically-principled conditioning* to generate point clouds from images, yielding better reconstructions than with unstructured global embeddings, with state of the art performance.
3. We bring denoising diffusion models for point cloud synthesis to the real world by applying our method to the Taskonomy dataset [243].

4.3 Related work

Denoising diffusion models. Denoising diffusion models [210, 80] are trained to denoise data perturbed by Gaussian noise. This process is applied iteratively during inference, and models are able to generate high-quality samples mirroring the distribution of the training data from random noise, optionally with a conditioning signal. They have shown great success synthesizing images from text [177, 187, 192], speech [30, 108, 168, 29], 3D objects [92, 166], and recently point clouds [135, 250, 244]. Diffusion models can be applied discretely, with a Markov chain [210, 80], or continuously with stochastic differential equations [105, 211]. We use a continuous formulation first proposed in [211], specifically an extension proposed in [98].

Generative point clouds models. Point cloud synthesis has been tackled with a wide array of techniques, including Variational Auto-Encoders (VAEs) [104], Generative Adversarial Networks (GANs) [232, 2, 119, 34, 207, 85, 104, 122], and autoregressive models [215]. Set-VAE [104] proposed an attention-based hierarchical VAE applicable to sets, such as point clouds. Achlioptas *et al.*[2] introduced l-GAN, operating over latents encoding shape, and r-GAN, directly on point clouds. SP-GAN [122] guides the generator with a global, uniformly distributed spherical prior and a local, random latent code to disentangle global and local shape. PointGrow [215] relies on an autoregressive model that samples each point conditionally on previously-generated points. ShapeGF [24] learns distributions over gradient fields, moving randomly sampled points to high-density areas such as surfaces. Xie *et al.*[235] formulate a permutation-invariant energy-based model with a PointNet. Of more direct importance to us are two other families, discussed separately: those based on normalizing flows (NF) and denoising diffusion models (DDM).

NFs for point cloud synthesis. Normalizing flows make powerful generative models and have been applied to point cloud synthesis [236, 103, 170, 107, 169]. PointFlow [236] broke ground by proposing a framework dividing the problem into two stages. First, a latent code responsible for the shape of the object, $s \sim P_{\theta_s}$, is sampled. Second, individual points p_i are sampled i.i.d. conditionally on s, meaning that a cloud $\{p_i\}$ is sampled with probability

$$P(\{p_i\}) = \int_{s \in \mathscr{S}} P_{\theta_s}(s) \prod_{i=0}^{n} P_{\theta_p}(p_i|s). \tag{4.1}$$

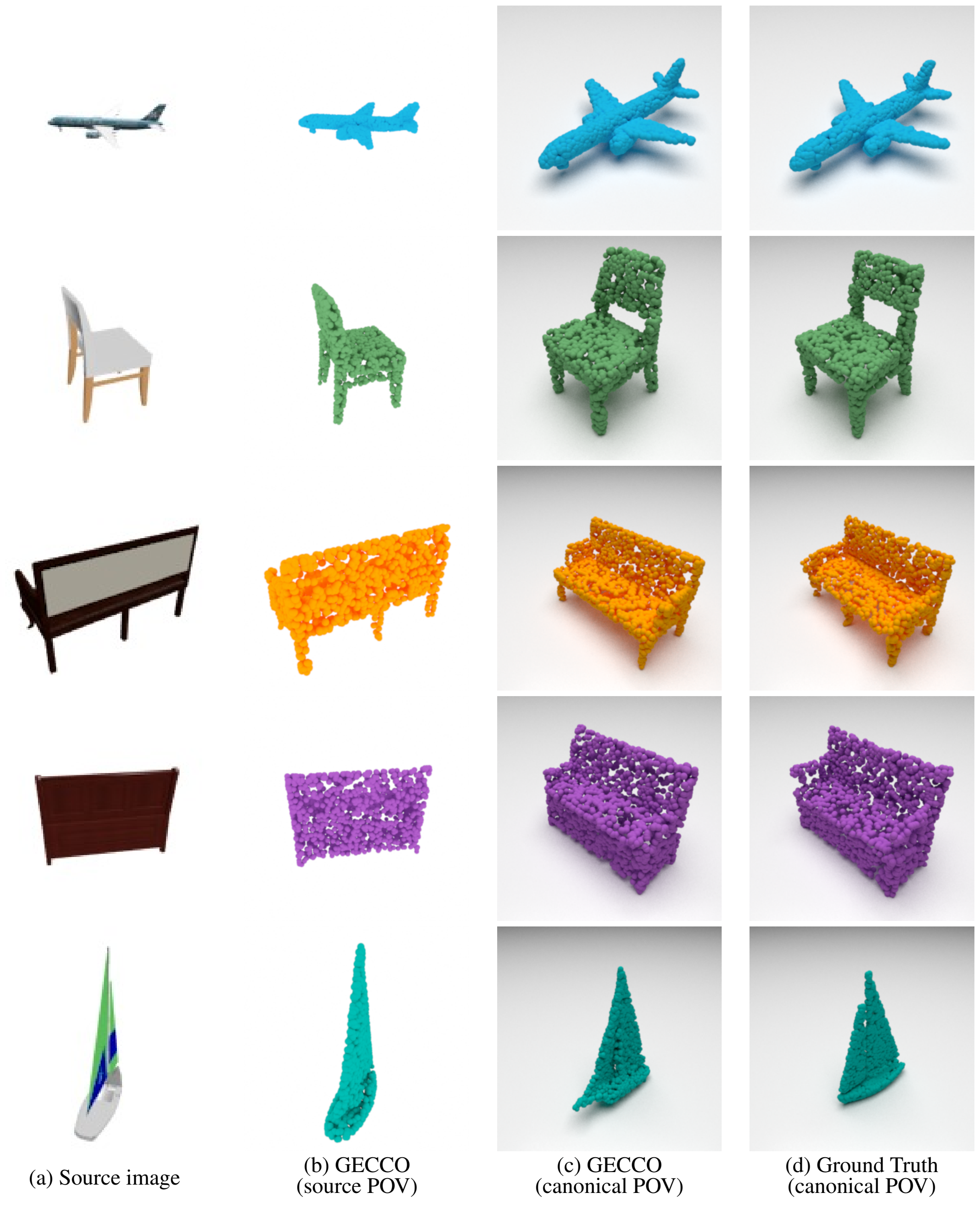


Figure 4.2: **Image-conditioned generation.** We condition denoising diffusion models with images (a) in a geometrically-principled manner. Our model can reconstruct the shapes accurately from that view (b), and also generate plausible completions for regions not visible in the image (c, d), even under drastic occlusions (rows 3-5).

A downside of this formulation is the intractability of the probability computation, requiring an integral over all s: they thus train with an ELBO loss. In contrast, C-flow [170] models the point cloud jointly with a NF. This avoids the intractable probability, but they encounter difficulty in defining invertible layers which respect the permutation equivariance of point clouds. Their solution is to canonicalize the order of points with space-filling curves, which makes the approach complex. An alternative would be to use continuous-time NFs [69]. Unfortunately, powerful continuous-time models are known to be slow and expensive to train [99]: as training progresses and the dynamics of the ODE become more complex, the cost of solving it with the precision necessary for stable backpropagation becomes impractically large.

DDMs for point cloud synthesis. This family, which our approach belongs to, has seen significant developments over the past year [135, 250, 244]. DPM [135] revisited the PointFlow [236] formulation (Eq. 4.3), replacing the normalizing flow $P(p_i|s)$ with a diffusion model. It relies on shape latents to parameterize the prior distribution with NFs, and uses them to condition a discrete DDM. It splits the loss into two additive terms, which requires tuning hyperparmeters – our approach is simpler. PVD [250] trains a discrete DDM directly on point clouds (without shape latents) using a point-voxel network (PVCNN) that enables 3D convolutions [130]. It can optionally take in depth images as input to perform shape completion on occluded regions. This is achieved by freezing a set of points, extracted from the depth map, and optimizing over a set of 'free' points – we do geometrically-principled conditioning with RGB images instead, which is more widely applicable. Moreover, the authors argue that conventional permutation-equivariant architectures operating on pure point representations such as PointNet++ [172] are difficult to apply to diffusion models – we show we can achieve similar performance with a very simple architecture [116]. In a different direction, PDR [139] proposes a dual-network approach to shape completion based on DDMs.

More recently, LION [244] proposed another two-stage approach. First, VAEs are used to obtain latent representations of both global shape, and points. Second, a DDM is trained to model those latent spaces. They use a continuous diffusion model, like we do, but their reliance on shape latents means that computing exact probabilities is not tractable. Like PVD, LION uses PVCNN [130] for the encoder, decoder, and diffusion models. Finally, it may condition samples with different signals, such as images or text embeddings, by conditioning the shape latent with adaptive Group Normalization in the PVCNN layers. In contrast, we use a convolutional backbone to extract image features at the locations 3D points project to, concatenate them to the positional features, and feed them to the network.

Other single-view reconstruction approaches. Most of the generative point cloud methods our approach belongs to tackle only the unconditional problem. There is a wide array of relevant works on shape synthesis from single images, including regression and generative models, and using different representations such as voxels or meshes. 3D-R^{2}N^2 [36] maps multiview images to occupancy grids with recurrent networks, but its resolution is limited due to using voxel grids.

Pixel2Mesh [227] produces meshes from images by progressively deforming an ellipsoid using intermediate features extracted from the image. AtlasNet [70], also mesh-based, generates 3D shapes by mapping multiple squares to shape surfaces. Chen *et al.*[28] learn point clouds from images by supervising their projections onto the image plane with samples from ground-truth silhouettes. Pix2Point [117] uses a 2D-3D hybrid network with an optimal transport loss to reconstruct point clouds from outdoor images. PSGN [59] combines an image and a random vector with a feedforward network to turn them into a point cloud, supervising with a permutation-equivariant loss. OccNet [143] tackles 3D reconstruction as the decision boundary of a learned occupancy classifier, which unlike voxel-based methods can be evaluated at arbitrary resolutions. Finally, there is of course a large body of work on monocular depth estimation: we refer the reader to [148].

4.4 Method

In order to follow the framework of [98, 211], we need to design a network s_θ to approximate the score $s_\theta(\mathbf{p}, t, c) \approx \nabla_\mathbf{p} \log p_t(\mathbf{p}|c)$, where c is an optional conditioning signal. We treat the point set $\{p_i\}, p_i \in \mathbb{R}^D, i = 1, \ldots, N$ as a vector $\mathbf{p} \in \mathbb{R}^{N \times D}$. Since our network s_θ is permutation-equivariant, working with $\mathbf{p}$ is sound. We present experiments with $D = 3$, but the approach is general.

4.4.1 Score network

Our score network $s_\theta(\mathbf{p}, t, c)$ (where dependency on c is optional) is inspired by the Set Transformer [116], which we choose as a powerful permutation-invariant architecture specifically designed for unordered inputs, such as point clouds. It treats each point as a token, but due to the quadratic scaling of self-attention, it instead uses cross-attention with a number of *inducers*, whose initial values are learned. Compared to the original Set Transformer, in each layer we use an extra shallow MLP on the inducers and not just on the tokens. Specifically to the task of diffusion, we "inject" the noise parameter t through the bias and scale of the Group Normalization [233] layers in the network, similarly to [244]. We also follow the recent approach of [176] and use Gaussian activations, allowing us to use a simple linear projection of input coordinates $\mathbb{R}^3 \rightarrow \mathbb{R}^{d_{\text{nn}}}$, where d_{nn} is the dimensionality of the input to the Set Transformer. We found this approach substantially better than the more common Fourier feature embedding.

4.4.2 Image-based conditioning

We undertake the goal of geometrically-principled point cloud generation conditioned on images. Specifically, we wish to: (a) generate point clouds in the reference frame of the camera; (b) accurately reconstruct the visible part of the object; and (c) build plausible hypotheses for occluded or ambiguous regions in a generative fashion. Note that the last property sets our approach apart from most supervised approaches, such as monocular depth, which treat the problem point/pixel-wise and largely ignore the different modes of the posterior. Property (a)

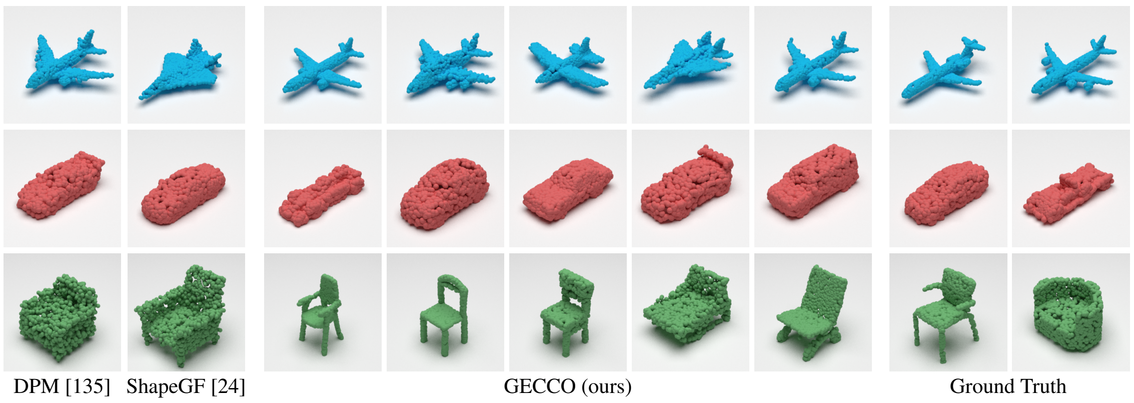


Figure 4.3: **Unconditional point cloud synthesis on ShapeNet.** All examples contain 2048 points.

differs from most prior work on conditional generative models for point clouds, which usually condense the image into an unstructured, global embedding, losing much of the geometric detail. We achieve our goals through *camera frustum reparameterization* of the point cloud and *point-wise projective conditioning*.

Projective conditioning. In order to provide the network with a highly accurate conditioning signal we need to maintain a geometrical interpretation of the model. We do so by using a ConvNeXt backbone [131] to extract multi-scale image features. At any time t in the diffusion process, we take the point cloud $\{p_i\}_t$, project the points onto the image plane, look up the image features corresponding to each projection using bilinear interpolation, and concatenate them to the point's location before feeding them to the transformer. This way each point knows the individual image properties *at its location*. We use this approach, which has a small computational overhead, to condition the reverse diffusion process *at every step*, as shown in Fig. 4.1. We concatenate the point location and its associated feature prior to the MLP that projects the features to d_{nn} dimensions, and use the exact same transformer architecture for conditional and unconditional settings, making this the only difference between $s_\theta(\mathbf{p}, t)$ and $s_\theta(\mathbf{p}, t, c)$. For points outside of image bounds the bilinear look-up simply returns zeros.

Camera frustum reparameterization. Current diffusion models are constrained to well-centered small objects and thus work directly in $\mathbb{R}^n$, but in Sec. 4.5.3 we wish to model points contained in the camera's viewing frustum only. Our solution is to *reparameterize* the coordinates. Since the projection of each point on the image plane $(p_h, p_w) \in [0,1]^2$, and its depth $p_d \in \mathbb{R}^+$, we can biject it to

$$(u, v, l) = \left(S^{-1}(p_h), S^{-1}(p_w), \log(p_d)\right) \in \mathbb{R}^3, \tag{4.2}$$

where S^{-1} is the inverse sigmoid function. We use standard diffusion in (u, v, l) and map points back to (x, y, z) at the end. We do not use any reparameterization for ShapeNet.

Implementation details Our Set Transformer-based network has 6 layers and takes inputs of dimensionality $d_{\text{nn}} = 384$. For unconditional models, we simply project the 3D point location to d_{nn}. For image-conditioned models, we extract multi-scale ConvNeXt features of sizes 96, 192, 384 (total: 672), concatenate them to the point's location, and project them to d_{nn}. We train our models with 1024 or 2048 points, subsampling datasets which contain more points per example, for data augmentation. We follow [97] in initializing transformer skip-connections with small weights and optimise the network (including the ConvNeXt) with AdaBelief [252] with learning rate $2 \cdot 10^{-4}$, for a number of steps depending on the dataset (see appendix). For each dataset we scale the data globally to zero mean, unit variance (which we undo at inference time) and pick $\sigma_{\max}$ by estimating the maximum pairwise distance between training examples. We train using the preconditioning and loss formulation of [98], with the main departure in that we sample σ values log-uniformly over $(10^{-4}, \sigma_{\max})$ instead of log-normally. As in [211], we found it beneficial to apply an exponential moving average to the weights of our model: we use a rate of 0.999. We implement our software with JAX [19] and Equinox [102] and use Diffrax [101] for ODE solving. For inference we use the 2nd-order stochastic sampler of [98] (later referred to as 'SDE') or the probability flow of [211] (later: 'ODE'), with 128 steps. Please refer to Sec. 4.5.4 for an ablation study, and to Table 4.6 for computational details: our approach is both *faster* and *lighter* than comparable methods.

4.5 Experiments

We use ShapeNet [26] to evaluate unconditional point cloud synthesis in Sec. 4.5.1, and with image conditioning in Sec. 4.5.2. We then show that our method can translate to larger-scale, real data on the Taskonomy dataset [243] in Sec. 4.5.3, and ablate it and showcase some of its properties in Sec. 4.5.4. We render point clouds with Mitsuba 3 [93].

4.5.1 Unconditional generation on ShapeNet

Dataset We evaluate our approach on the dataset most commonly used for generative shape modelling: ShapeNet [26]. We follow the methodology, splits, and metrics introduced by PointFlow [236], which provides point clouds sampled from the original meshes, and train single-class models for three categories: airplane, chair, and car. In order to ensure reproducibility we compare against the results published in [244], the most recent and thorough. While PointFlow normalizes the data globally to zero-mean per axis, and unit variance, others methods use per-shape normalization: we consider both. We use 2048 points for all methods.

Metrics We consider two distance metrics between point clouds: the chamfer distance (CD), which measures the average squared distance between each point in one set to its nearest neighbor on the other set; and the earth mover's distance (EMD), which solves the optimal transport problem. Given point sets $\mathbf{p} = \{p_i\}$ and $\mathbf{q} = \{q_i\}$ with N points each, and ϕ a bijection between

		MMD↓		COV↑ (%)		1-NNA↓ (%)	
	Model	CD	EMD	CD	EMD	CD	EMD
AIRPLANE	Oracle	0.214	0.369	46.17	49.88	64.44	63.58
	r-GAN [2]	0.447	2.309	30.12	14.32	98.40	96.79
	l-GAN-CD [2]	0.340	0.583	38.52	21.23	87.30	93.95
	l-GAN-EMD [2]	0.397	0.417	38.27	38.52	89.49	76.91
	PointFlow [236]	0.224	0.390	47.90	46.41	75.68	70.74
	SoftFlow [103]	0.231	0.375	46.91	47.90	76.05	65.80
	SetVAE [104]	0.200	0.367	43.70	48.40	76.54	67.65
	DPF-Net [107]	0.264	0.409	46.17	48.89	75.18	65.55
	DPM [135]	0.213	0.572	48.64	33.83	76.42	86.91
	PVD [250]	0.224	0.370	48.88	52.09	73.82	64.81
	LION [244]	0.219	0.372	47.16	49.63	67.41	61.23
	GECCO (ours)	0.245	0.368	48.15	48.40	72.10	62.96
CHAIR	Oracle	2.618	1.555	53.02	51.21	51.28	54.76
	r-GAN [2]	5.151	8.312	24.27	15.13	83.69	99.70
	l-GAN-CD [2]	2.589	2.007	41.99	29.31	68.58	83.84
	l-GAN-EMD [2]	2.811	1.619	38.07	44.86	71.90	64.65
	PointFlow [236]	2.409	1.595	42.90	50.00	62.84	60.57
	SoftFlow [103]	2.528	1.682	41.39	47.43	59.21	60.05
	SetVAE [104]	2.545	1.585	46.83	44.26	58.84	60.57
	DPF-Net [107]	2.536	1.632	44.71	48.79	62.00	58.53
	DPM [135]	2.399	2.066	44.86	35.50	60.05	74.77
	PVD [250]	2.622	1.556	49.84	50.60	56.26	53.32
	LION [244]	2.640	1.550	48.94	52.11	53.70	52.34
	GECCO (ours)	2.793	1.601	46.68	49.40	56.57	54.68
CAR	Oracle	0.938	0.791	50.85	55.68	51.70	50.00
	r-GAN [2]	1.446	2.133	19.03	6.539	94.46	99.01
	l-GAN-CD [2]	1.532	1.226	38.92	23.58	66.49	88.78
	l-GAN-EMD [2]	1.408	0.899	37.78	45.17	71.16	66.19
	PointFlow [236]	0.901	0.807	46.88	50.00	58.10	56.25
	SoftFlow [103]	1.187	0.859	42.90	44.60	64.77	60.09
	SetVAE [104]	0.882	0.733	49.15	46.59	59.94	59.94
	DPF-Net [107]	1.129	0.853	45.74	49.43	62.35	54.48
	DPM [135]	0.902	1.140	44.03	34.94	68.89	79.97
	PVD [250]	1.077	0.794	41.19	50.56	54.55	53.83
	LION [244]	0.913	0.752	50.00	56.53	53.41	51.14
	GECCO (ours)	1.044	0.769	50.00	56.82	56.82	49.15

Table 4.1: **Unconditional generation (global normalization).** Generation metrics on three ShapeNet categories. MMD-CD is multiplied by 10^3, and MMD-EMD by 10^2. The top 3 are highlighted in gray (darker is better).

		MMD↓		COV↑ (%)		1-NNA↓ (%)	
	Model	CD	EMD	CD	EMD	CD	EMD
AIRPLANE	Oracle	0.230	0.539	42.72	45.68	69.26	67.78
	TreeGAN [207]	0.558	1.460	31.85	17.78	97.53	99.88
	ShapeGF [24]	0.313	0.636	45.19	40.25	81.23	80.86
	SP-GAN [122]	0.403	0.766	26.42	24.44	94.69	93.95
	PDGN [85]	0.409	0.701	38.77	36.54	94.94	91.73
	GCA [245]	0.359	0.765	38.02	36.30	88.15	85.93
	LION [244]	0.356	0.593	42.96	47.90	76.30	67.04
	GECCO (ours)	0.354	0.572	44.20	50.12	76.17	68.89
CHAIR	Oracle	3.864	2.302	49.7	42.11	55.14	54.76
	TreeGAN [207]	4.841	3.505	39.88	26.59	88.37	96.37
	ShapeGF [24]	3.724	2.394	48.34	44.26	58.01	61.25
	SP-GAN [122]	4.208	2.620	40.03	32.93	72.58	83.69
	PDGN [85]	4.224	2.577	43.20	36.71	71.83	79.00
	GCA [245]	4.403	2.582	45.92	47.89	64.27	64.50
	LION [244]	3.846	2.309	46.37	50.15	56.50	53.85
	GECCO (ours)	4.119	2.410	48.64	52.42	55.36	56.80
CAR	Oracle	1.05	0.829	47.44	48.01	57.53	56.68
	TreeGAN [207]	1.142	1.063	40.06	31.53	89.77	94.89
	ShapeGF [24]	1.020	0.824	44.03	47.16	61.79	57.24
	SP-GAN [122]	1.168	1.021	34.94	31.82	87.36	85.94
	PDGN [85]	1.184	1.063	31.25	25.00	89.35	87.22
	GCA [245]	1.074	0.867	42.05	48.58	70.45	64.20
	LION [244]	1.064	0.808	42.90	50.85	59.52	49.29
	GECCO (ours)	1.063	0.802	46.31	49.15	60.51	47.87

Table 4.2: **Unconditional generation (per-shape normalization).** Same as Table 4.1, with per-shape normalization.

	3D-R^{2}N^2	PSGN	Pixel2Mesh	AtlasNet	OccNet	GECCO	OccNet w/ ICP	GECCOw/ ICP
airplane	0.227	0.137	0.187	**0.104**	0.140	0.106	0.151	**0.081**
bench	0.194	0.181	0.201	0.138	0.157	**0.097**	0.158	**0.088**
cabinet	0.217	0.215	0.196	0.175	0.156	**0.110**	0.141	**0.100**
car	0.213	0.169	0.180	0.141	0.153	**0.103**	0.139	**0.093**
chair	0.270	0.247	0.265	0.209	0.209	**0.142**	0.196	**0.117**
display	0.314	0.284	0.239	0.198	0.260	**0.138**	0.247	**0.119**
lamp	0.778	0.314	0.308	0.305	0.394	**0.186**	0.380	**0.164**
loudspeaker	0.318	0.316	0.285	0.245	0.269	**0.158**	0.251	**0.141**
rifle	0.183	0.134	0.164	0.115	0.142	**0.097**	0.155	**0.073**
sofa	0.229	0.224	0.212	0.177	0.185	**0.123**	0.188	**0.114**
table	0.239	0.222	0.218	0.190	0.176	**0.107**	0.207	**0.111**
telephone	0.195	0.161	0.149	0.128	0.129	**0.090**	0.138	**0.083**
vessel	0.238	0.188	0.212	0.151	0.200	**0.132**	0.203	**0.128**
average	0.278	0.215	0.216	0.175	0.198	**0.122**	0.196 (+1.02%)	**0.109** (+11.9%)

Table 4.3: **Image-conditional generation on ShapeNet-Vol.** L1 Chamfer distance between samples reconstructed from an image and the ground truth point clouds (lower is better), following [143]. Qualitative results are available in Fig. 4.2.

them, they are defined as:

$$\mathrm{CD}(\mathbf{p},\mathbf{q}) = \frac{1}{N}\left[\sum_{p\in\mathbf{p}}\min_{q\in\mathbf{q}}\|p-q\|_2^2 + \sum_{q\in\mathbf{q}}\min_{p\in\mathbf{p}}\|p-q\|_2^2\right], \tag{4.3}$$

$$\mathrm{EMD}(\mathbf{p},\mathbf{q}) = \frac{1}{N}\min_{\phi:\mathbf{p}\to\mathbf{q}}\sum_{p\in\mathbf{p}}\|p-\phi(p)\|_2. \tag{4.4}$$

Given these two similarity functions, we sample as many point clouds as there are in the reference set $\mathbf{S_r}$ to obtain a generated set $\mathbf{S_g}$ and compute three metrics between the ground truth and sampled collections. To compute **coverage (COV)** we find the nearest neighbor in the reference set for each point cloud in the generated set, and compute the fraction of shapes in the reference set that are matched to at least one shape in the reference set. It can capture mode collapse, but not the quality of the samples. The **minimum matching distance (MMD)** is a complementary metric that measures the average minimum distance from every sample in the reference set to every sample in the generated set. PointFlow [236] proposes an arguably better metric also used for GANs: **1-Nearest Neighbour Accuracy (1-NNA)**. It is defined as the accuracy of a leave-one-out classifier that assigns each sample in $\mathbf{S_r} \cup \mathbf{S_g}$ to the 'class' (set) of its closest neighbor, other than itself. Note that a perfect oracle would score ~50%. As all three metrics rely on nearest neighbours, they can be computed with CD or EMD: we report both. Details are provided in the supplementary material.

Results Results are shown in Tab. 4.1 for global normalization, and Tab. 4.2 for per-shape normalization. 1-NNA is the metric favored by most recent papers. Our method performs on par with LION, the state of the art, on the first benchmark and slightly outperforms it on the second. In addition to the baselines, we consider an oracle that spits out samples from the training set instead of generating novel ones. Our method outperforms this oracle in 1-NNA-EMD for all three categories in Tab. 4.1, which suggests that the dataset is at the saturation point, or that the distance metrics fail to fully capture the quality of the samples. We show samples and ground truth examples in Fig. 4.3.

4.5.2 Conditional generation on ShapeNet-Vol

Dataset. None of the baselines used in the previous section are able to condition the generative process with images, with the exception of LION, which may optionally train DDMs conditioned with CLIP embeddings [175, 196] extracted from ShapeNet renders. While seemingly effective this approach is not geometrically principled, and the paper offers only qualitative examples. We thus turn to the ShapeNet-Vol benchmark, originally introduced by 3D-R^{2}N^2 [36], which provides rendered images and voxelized models for 13 ShapeNet categories: each shape is rendered from 24 viewpoints at 137×137 pixels. We align the point clouds to the camera pose for each view and train our models with the conditioning scheme of Sec. 4.4.2. Note that unlike the unconditional experiments in the previous section, here we train *a single model for all 13 categories*.

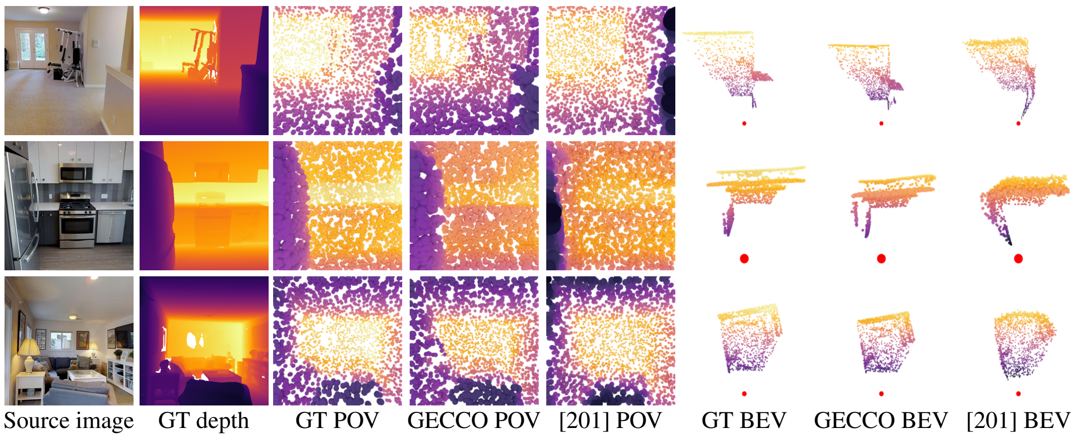


Figure 4.4: **Qualitative examples on Taskonomy.** Color encodes depth. We show point clouds with 2048 points each, from the cameras's point of view, and from a bird's eye view. The red dot in BEV marks the location of the camera.

Results. We follow the evaluation protocol for single-view reconstruction introduced by OccNet [143]. For mesh-based methods, such as OccNet, the benchmark samples 100k points from the mesh and computes the chamfer distance between the generated and ground truth point clouds as a quality metric – including points occluded in the image. For point-based methods such as PSGN [59], the benchmark simply samples more points (no meshing). In order to reach the required number of points, we simply generate multiple samples for each image and concatenate them. Note that following [143], we use the *L1 chamfer distance*, defined as in Eq. 4.3 but without squaring the norms. OccNet reconstructs the model in a canonical reference frame, which is also used for evaluation, but our method generates predictions from the point of view of the camera: we move our predictions to this canonical frame with the ground truth pose. We report results in Tab. 4.3[1]. Our models outperform all mesh-based methods, including OccNet, and also PSGN. Additionally, we noticed that our generated point clouds were slightly misaligned, but not those from OccNet. We hypothesized this was due to OccNet generating samples in a canonical reference frame rather than the camera's point of view, which while advantageous here does lose generalization. We confirmed this by aligning the point clouds with ICP [16, 178] and re-computing the metric: our method improves by 12% relative, compared to OccNet's 1%. Note that unlike OccNet, our approach does not need normalized data in a canonical pose and it can deal with non-watertight meshes. On the other hand, it does require known camera intrinsics. We show qualitative examples in Fig. 4.2.

4.5.3 Conditional generation on Taskonomy

We also wish to showcase how our method scales to real data, and past the object-centric nature of ShapeNet that most generative methods are limited to. For this purpose we turn to

[1]For OccNet we run the latest model available in their repository, improved from the original paper. For others we use the results from [143].

Split	Model	Chamfer ↓	Chamfer (ICP) ↓
Test	GECCO	0.661 / **0.502**	**0.444 / 0.242**
	Monocular depth [201]	**0.632** / 0.527	0.558 / 0.451
Val	GECCO	**0.541 / 0.427**	**0.361 / 0.222**
	Monocular depth [201]	0.567 / 0.490	0.497 / 0.418

Table 4.4: **Evaluation on Taskonomy** with mean / median values. Note that we do not use the validation set for early stopping, so ‘validation’ and ‘test’ both act as test sets. The difference in performance in ‘test’ is due to out-of-distribution scenes in that subset (see appendix for details).

Taskonomy [243], which contains a large dataset of scanned indoor scenes with high-quality depth maps, and convert them into point clouds by sampling and unprojecting 8192 points per image. We sample points inversely proportionally to pixel depth, to emulate per-surface-area densities. This yields a rich image-point cloud dataset.

Given the lack of generative methods that can scale up to this data, we compare with a monocular depth method from [201], also trained on Taskonomy. We first adjust the absolute scale and shift of its output by comparing with ground truth depth (as in the loss function of MiDaS [114]) and proceed by unprojecting with the same procedure as when creating the dataset. For GECCO, we directly predict the point clouds in absolute units, using the (u, v, l) reparameterization introduced in Sec. 4.4.2. We use 2048 points for both training and evaluation. We compare the two approaches qualitatively in Fig. 4.4, and quantitatively in Tab. 4.4, using the same metric as in Sec. 4.5.2 and Tab. 4.3. This experiment confirms our method extends beyond object-centric views to real scenes, and greatly outperforms similarly-sized baselines benefitting from years of research on monocular depth. As in Sec 4.5.2, we also report results with ICP. For the baseline we disabled scale estimation, as it degrades the results.

4.5.4 Ablation studies and further experiments

Ablation study: global vs projective conditioning. We evaluate our approach to conditioning the denoising process through projective geometry with the more standard approach relying on a global embedding. Instead of bilinear lookup for each point, we mean-pool the CNN features and inject them globally as in [244], alongside t, through the bias and scale of normalization layers. We compare both approaches on the OccNet benchmark of Sec. 4.5.2. As we do not aim to compare against those baselines, we use 1024 points rather than 100k, for simplicity. Results are shown in Tab. 4.5. Our projective conditioning is 7-8% more accurate.

Ablation study: numerical solvers. We consider the OccNet [143] benchmark and ablate the use of the probability flow solver proposed in [211] versus the stochastic solver used in [98]. We find the latter superior and turn to the effect of the number of solver steps. We observe that more solver steps bring larger perceptual improvements which are not always captured by the chamfer distance, up to about 128 steps, which we use in all other experiments in the paper. We report

Conditioning	Sampler	N_{denoise}	ICP	Subset	L1-CD↓	Δ↑ (%)
Global	ODE	128	✗	–	0.305	–
Projective	ODE	128	✗	–	0.286	+6.6%
Global	SDE	128	✗	–	0.302	–
Projective	SDE	128	✗	–	0.283	+6.7%
Global	ODE	128	✓	–	0.280	–
Projective	ODE	128	✓	–	0.259	+8.1%
Global	SDE	128	✓	–	0.276	–
Projective	SDE	128	✓	–	0.257	+7.4%
Projective	SDE	8	✗	–	0.615	-117.3%
Projective	SDE	16	✗	–	0.309	-9.2%
Projective	SDE	32	✗	–	0.286	-1.1%
Projective	SDE	64	✗	–	0.287	-1.4%
Projective	SDE	128	✗	–	0.283	–
Projective	SDE	8	✓	–	0.353	-37.4%
Projective	SDE	16	✓	–	0.267	-3.9%
Projective	SDE	32	✓	–	0.258	-0.4%
Projective	SDE	64	✓	–	0.258	-0.4%
Projective	SDE	128	✓	–	0.257	–
Global	SDE	128	✗	50%	0.308	-2.0%
Global	SDE	128	✓	50%	0.283	-2.5%
Projective	SDE	128	✗	50%	0.289	-2.1%
Projective	SDE	128	✓	50%	0.261	-1.6%

Table 4.5: **Ablation study on the OccNet benchmark.** We compare conditioning with global context vs projective lookups (sec. 4.4.2), probability flow (‘ODE’) [211] vs the stochastic solver (‘SDE’) of [98], the number of solver iterations, and the impact of reducing the training data to 50%. We conduct these experiments with 1024 points, for speed.

	Num. params	Inference speed
LION (unconditional)	110M	2.51 s/example
PVD (unconditional)	27.7M	2.95 s/example
GECCO (unconditional)	13.7M	0.25 s/example
GECCO (conditional)	47.7M	0.27 s/example

Table 4.6: **Size and speed comparison.** Measured on an NVIDIA A100 GPU with 40Gb, with 2048 points and batch of 64. GECCO uses the SDE sampler with 128 steps.

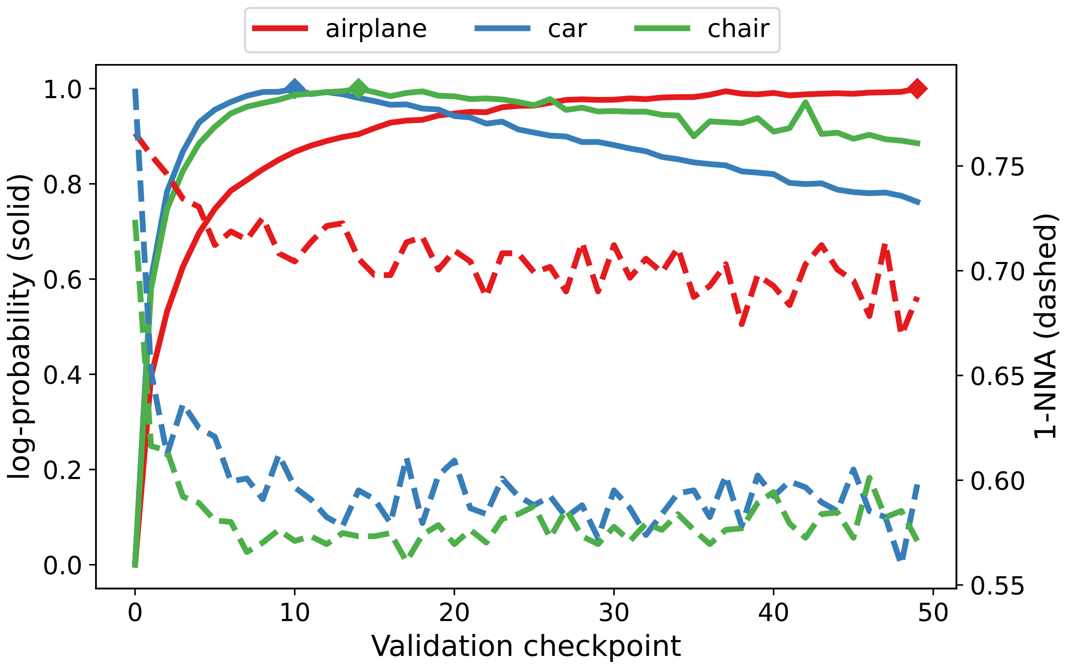


Figure 4.5: **Log-probabilities while training.** Dashed line: CD 1-NNA (Sec. 4.5.1) on the validation set. Solid: validation set log-probability (normalized); diamonds mark maxima. Notice how the decreasing likelihood of sampling validation examples does not increase (*i.e.*, degrade) 1-NNA.

the numbers in Tab. 4.5: relative values take $N_{denoise} = 128$ as reference. It should be noted that technically, both solvers use two network evaluations per step.

Ablation study: training with fewer samples. We train conditional models with only 50% of the data and report a surprisingly small drop in performance: ~2% (Tab. 4.5). This holds for both projective and global conditioning.

Probabilities as a metric. Our approach allows us to compute exact probabilities over the validation set, following the method in [211]. The maxima in probability corresponds to the optimal state in terms of *novel* generative performance. On ShapeNet, which is quite small, we notice that our models overfit in terms of this metric while maintaining low 1-NN accuracy: we report their evolution in the unconditional setting in Fig. 4.5. This corroborates our observation that the ShapeNet benchmark is relatively saturated: by some metrics, state-of-the-art methods may even outperform an oracle which simply returns the training set. We argue that tractable likelihoods may prove very useful in datasets and tasks with no other easy means of validation.

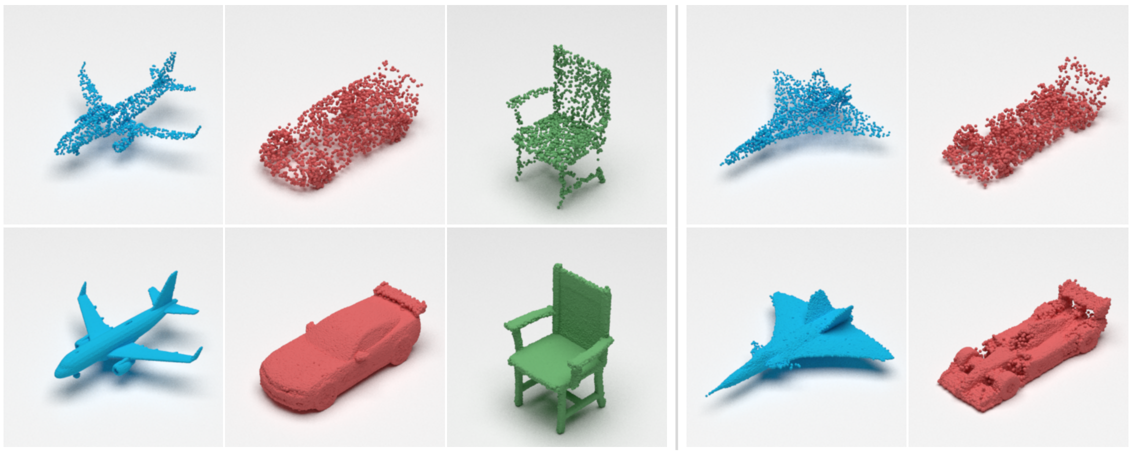


Figure 4.6: **Point cloud upsampling examples.** Top: original point clouds with 2048 points. Bottom: we upsample them by 50x to 102k points. Left: Our inpainting technique yields high-quality point clouds. Right: Artifacts may appear occasionally, especially on complex, irregular structures. Note: figure is rendered with a smaller point size.

Point cloud upsampling by inpainting. Another application of our method is point cloud upsampling. We first discuss a naive scheme, directly following the blueprint of [134] which uses this technique to inpaint image. To upsample from b to u points (that is generate $(u-b)$ new ones) we diffuse the input $\{p_{0..b}\}$ to $\sigma_{\max}$ and concatenate with new points sampled from the prior, $\{p_{b..u}\}$. We then reverse the diffusion, computing the scores on all points $\{p_{0..u}\}$, but using them only to update $\{p_{b..u}\}$, while $\{p_{0..b}\}$ is reversed deterministically to the input. Compared to models following Eq. 4.3, this approach treats the input $\{p_{0..b}\}$ as the latent code s, albeit the new points $\{p_{b..u}\}$ are not conditionally independent. We use 4 resampling sub-steps (see [134]) per solver step, and when upsampling by large factors, in order to stay in the range of u the network is trained for, we concatenate multiple conditionally-independent completions of $\{p_{0..b}\}$. We find this procedure to result in coherent, high-resolution point clouds: see Fig. 4.6 for examples.

Faster and more accurate upsampling with Set Transformer. A more efficient technique exploits the properties of the Set Transformer [116] backbone of GECCO. When employing the naive algorithm, for optimal performance it is preferable to keep the network's input distribution as close as possible to that at training time. This means that ideally we would want $u = b+1$, where b and u are the cardinality of the original and upsampled point clouds, respectively. This way a cloud can be upsampled to u points, by repeating that procedure $(u-b)$ times and concatenating the results. This however is as expensive as generating a cloud of $b+1$ points $(b-u)$ times from scratch. With GECCO's complexity being linear in the size of the processed cloud, the total cost is $O(b(u-b))$.

Fortunately, we can bring this cost down to approximately that of sampling a cloud of u points once. In Set Transformer the points do not interact directly with each other, but rather via the inducers (see Sec. 4.4.1 and [116] for definition). If we make the assumption that with many points the influence of any one on the inducers is negligible, we can reverse-diffuse many new (inpainted) points *in parallel*, sharing the inducer state. This leads to a simple algorithm starting from the conditioning set $\{p\}_{0..b}^{\sigma=0}$ and the pure noise inputs $\{p\}_{b..u}^{\sigma=\sigma_{\max}}$. At each step t of reverse diffusion:

1. The conditioning input is diffused to σ_t to obtain $\{p\}_{0..b}^{\sigma_t}$.
2. The score network is evaluated on $\{p\}_{0..b}^{\sigma_t}$. We cache the activations of the inducers across the network's layers and discard the score estimate.
3. The score network is ran again, this time on $\{p\}_{b..u}^{\sigma_t}$, but with the inducer activations provided by the cache from point 2. We obtain the score for the inpainted points $b..u$.
4. We update $\{p\}_{b..u}$ as usual, using the score from point 3.

Not only does this avoid recomputing the inducer state for each individual point, bringing the cost from $O(b(u-b))$ to $O(u)$. The use of cache in step 3. means that for the newly sampled points we do not have to evaluate the attention in direction point $\rightarrow$ inducer but only inducer $\rightarrow$ point, additionally saving computation. The effectiveness of technique makes it practical to upsample using all b input points for conditining, increasing the strength of that signal and the quality of outcomes. Unlike the naive algorithm, first sampling the low resolution point cloud $\{p\}_{0..b}$ followed by sampling $\{p\}_{b..u}$ using this improved version is exactly equivalent to the model of Eq. 4.3 if we treat $\{p\}_{0..b}$ as the latent shape s and $\{p\}_{b..u}$ as the actual output.

4.6 Conclusions

We propose a novel approach to condition denoising diffusion models in a geometrically-principled manner by projecting generated point clouds to an image and augmenting point locations with sparse features from a convolutional backbone. Our framework relies on a simple permutation-equivariant transformer, trained with a continuous-time diffusion scheme. It yields state-of-the-art results in single-view synthesis, while performing on par in the unconditional setting. It can also deliver exact probabilities, and upsample by inpainting. We believe this is a first step towards controllable diffusion point cloud models on real data. Future work will explore occlusion on large-scale datasets, multi-view inference, and completion via inpainting.

4.7 Background

The original goal for GECCO was to advance density estimation on point clouds. If evaluated efficiently it can be used as probabilistic prior on 3D reconstructions from image matches, for example those generated with DISK (Sec. 2), helping regularize the process and reject errors. Unfortunately, the diversity of real world scenes makes it not feasible to design such a prior by hand, requiring a learned solution. Learning to estimate the continous probability density $\hat{P}(x) \approx P(x)$ from a set of discrete samples (exemplars) is notoriously difficult due to intractability of the partition function $z = \int_{x \in X} \hat{P}(x)$ for a general class of models $f : x \to \mathbb{R}$. On the other hand, many commonly used classes of functions with tractable z are either too simple to capture the diversity of real world point clouds or do not respect their unique properties, in particular permutation equivariance. For this reason, starting in summer 2021, we kept an eye on the developments in this field and created GECCO when the goal of robust density estimation on point clouds became feasible. We discuss the open problems on the way to practical use of such priors in the next section and here we focus on the progress made thus far. This section provides a literature review of the recent developments in modelling point clouds, extending the related work (Sec. 4.3) with special emphasis on density estimation.

(Discrete time) normalizing flows. The first expressive parametric density estimators, already in 2015, were normalizing flows [49, 184]. NFs reparametrize a normal prior with a series of operators designed to be highly expressive but constrained to invertibility and a tractable Jacobian. This makes the whole reparametrization invertible. To estimate a sample's probability under the model it is passed though this mapping in reverse (inverse), while accumulating the total Jacobian across the blocks. The resulting vector's probability under the prior and the total Jacobian of the reparametrization allow for computation of the sample's probability through the change of variables theorem. Still, applying normalizing flows to point clouds was difficult due to their permutation invariance. Early NFs treated the sample as a vector in $\mathbb{R}^n$ and operated on its dimensions in an arbitrary and asymmetrical fashion, which would to leave artifacts in point cloud structure. One attempt to circumvent this was made in C-Flow [170], which used a space-filling curve to canonicalize the order of points in a point cloud, but this solution was cumbersome for the purpose and delivered limited results.

Continuous-time normalizing flows. The second breakthrough happened in 2018, with Chen et al. [32] realizing that neural networks can be treated as operators in an ordinary differential equations, leading to higher-order models of form

$$f_{\text{ODE}}(x) = x + \int_0^1 f_\theta(x_t, t)dt \tag{4.5}$$

with $x_{t=0} = x$. The forward pass of neural ODE models is evaluated by plugging the network f_θ into a numerical solver and the backward pass by either backpropagating through the solver's internal operation or by solving an adjoint ODE. These choices trade off speed for memory

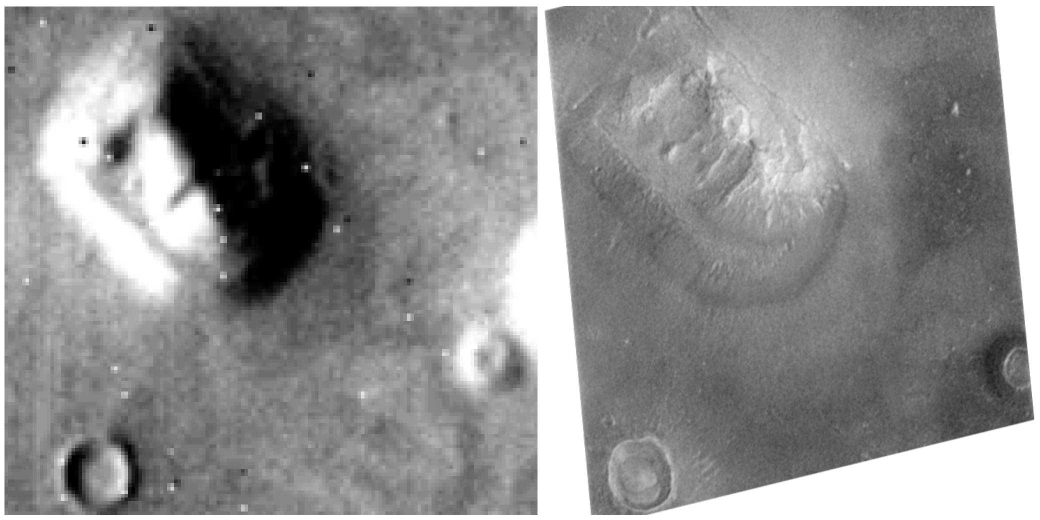


Figure 4.7: Denoising diffusion models train a network to clean noisy data examples. Such a network has only even "seen" data from the training distribution and will always return a plausible reconstruction, even given pure noise as input. This is similar to human brains which are wired to recognize faces to the point of seeing them even on Mars. Left: the famous "face on Mars" pictured by the Viking 1 orbiter. Right: the same area pictured by Mars Global Surveyor. Images from [157].

needed to keep track of network activations throughout the solver's operation. Aside from the theoretical importance of connecting general standard residual networks with ODEs, Chen's work had big practical implications for normalizing flows. Since an ODE defines a bijection between $x_{t=0}$ and $x_{t=1}$, a neural ODE can replace the chain of discrete reversible operators used in discrete-time normalizing flows as long as the Jacobian of f_{ODE} is efficient to compute. Although exact solutions are not usually available, a good stochastic estimator can be employed [86, 209] for results precise enough to train such continuous-time NFs. These pieces were put together in FFJORD [69], showing promising results on simple datasets such as CIFAR10 [109] and MNIST [115]. This was the first framework which allowed for integration of powerful permutation equivariant networks to modelling and density estimation of point clouds, bringing the *expressivity* needed for the task. Unfortunately, our experiments with CNFs for point cloud density estimation were stopped by lack of *scalability* — as the training progresses, the ODE trajectories become more complex and slower to solve, making the whole process impractically slow. This problem was discussed by contemporaneous literature [99] but without a promise of a breakthrough in our application. Still, CNFs were successfully used to model point clouds in a factorized formulation, precluding density estimation. This is discussed in Sec. 4.3.

Continous time diffusion models. The third and final breakthrough which enabled the development of GECCO was Song et al. making the connection between diffusion models (DMs) and CNFs in [211]. DMs are a family of generative models with history reaching back to 2015 [210] but which failed to show strong results until 2020 [80, 48]. The variants are discussed in depth in [41], here we summarize briefly. The operation of DMs relies on defining a Markov chain, called the (forward) diffusion process, which progressively corrupts data points until they are virtually indistinguishable from each other. A neural network is then trained to reverse this process step-wise, in least squares sense. Such a network usually has the signature

$$f_\theta(x,t) \in \mathbb{R}^n, x \in \mathbb{R}^n, t \in \mathbb{N} \tag{4.6}$$

where t informs the model about the time index in the Markov chain and thus the total magnitude of noise in x. Once trained, the network can be used to reverse the chain and "denoise" pure noise by iteratively applying

$$x_{t-1} = g(f(x_t,t),x_t,t) + h(t,\omega) \tag{4.7}$$

where g and h are two manually designed functions depending on the specific definition of the Markov chain and $\omega \in \mathbb{R}^n$ is a sample of random noise, independent across t. Overall, sampling with DMs is a random walk in the space of $\mathbb{R}^n$, biased by $f(x,t)$ towards regions of high data likelihood. Song's insight is similar to that of Chen et al. [32] — DMs are a time-discretized variant of a time-continous phenomenon described by a stochastic differential equation (SDE). They propose to train the networks for denoising at arbitrary, real-valued, noise levels and to use prior work on solving SDEs to obtian faster convergence and controllable speed-quality tradeoffs. They also use the theory of SDEs to show an associated ODE which uses the same network f_θ to define deterministic trajectories with marginals matching those of the SDE sampler.

Simulation-free training. Although not stated explicitly in the paper, the ODE formulation of CDM sampling is the same as for sampling from a CNF and it thus shows that both algorithm families are different ways to train the same probabilistic model. Crucially, CDM factorizes the training objective over individual network forward passes, as opposed to CNF which requires simulating the entire dynamics from the prior to data space. In that sense, to our best knowledge, Song et al.'s CDM were the first *simulation-free* training method for this class of probabilistic models, with more emerging recently, especially for specific domains [128, 31, 216]. Avoiding simulation has a twofold benefit. Firstly, it reduces the computational cost of learning from a single data point, as well as makes it deterministic, allowing for larger and more diverse batches. Secondly, it removes the problem of simulation precision and its interplay with gradient propagation, greatly reducing the surface area for optimization instabilities. These benefits are sufficient to allow the jump from plausible samples on MNIST and CIFAR10 with CNFs to realistic high resolution face pictures sampled from CDM in [211]. With the efficiency of training and the expressive power of CDM models, [211] was the paper which allowed the development of GECCO.

4.8 Limitations

Despite its success at learning distributions over point clouds, practical use of GECCO as a data-driven prior requires more work. The problems stem partially from issues with the data and partially with the diffusion model itself.

A sketch of a system which uses a learned prior to regularize stereo 3D geometries, as envisioned at the beginning of the project, is as follows. First a set of point clouds resulting from (unregularized) triangulation of feature matches in stereo image pairs is used to train the prior. Creation of such a dataset is relatively easy by taking sparse SfM models, such as those from MegaDepth [124], and picking subsets of points which are jointly registered to any given pair of images. After learning a density estimate of such point clouds $\hat{P}(x)$, one can integrate it directly into RANSAC hypothesis scoring or alternatively after RANSAC, to flag low likelihood solutions. A more sophisticated approach would involve a graphical probabilistic model jointly describing 3D landmark locations, relative camera positions (with their own prior) and in- and outlier matches. Finally, one can use a conditional variant of GECCO to additionally leverage the shape information present in the input views.

Data. The problem with building such a system starts with normalizing data. Stereo and multiview triangulation solves for the geometry up to scale. This ambiguity can in principle be ignored and left to the model to learn but in that case a large part of its expressive power will be spent on covering the scale of point clouds, rather than the shape and detail. An alternative is to pick some normalization factor (for example σ_{pc}^{-1}) and model such normalized point clouds. Unfortunately this makes the scale of each cloud very susceptible to outliers which may greatly impact the scaling factor, even in small numbers. This means that the network's expressiveness we intended to free by normalizing the point clouds will instead be used to model the few outlier points as they have by far the largest impact on the overall point cloud's shape. A third and the most principled solution would be to learn from a dataset with unified scales. This is already the case within a single SfM model, but to unify the scales across a dataset like MegaDepth would require in-person measurements at each world landmark or at least tedious geolocalization of the point cloud with high resolution maps. Otherwise any single scene is unlikely to be diverse enough for good generalization. This is the reason why in the paper we opted for the Taskonomy [243] dataset, as it contains diverse scenes in common scale (meters). It should be noted that the model trained and evaluated in the paper is not suitable as a prior for stereo triangulation as the learned point distribution is uniform on the image plane, as opposed to the spatial distribution of local features of choice (SIFT, DISK, etc). It would however be relatively straightforward to run SfM on the scenes in the dataset and then use the ground truth depth maps to unify their scale. To conclude, future work on the data aspect of GECCO will need to tackle the problems of unifying scale, dealing with outliers, and improving scene diversity.

Model. While developing GECCO we also uncovered some algorithmic issues with its application as a learned prior. First and most important is inference speed. Although GECCO is substantially faster for *sampling* than prior work (see Tab. 4.6), evaluation of density requires both network forward and backward pass at each of the ODE solver steps. This, compared to just forward passes for sampling, makes inference slow enough to be prohibitive in the inner loop of a RANSAC algorithm and problematic even as a final rejection check in pairwise matching of larger image collections. A secondary problem is with point cloud sizes. Empirically, GECCO trained with N points generalizes reasonably well in the range of $0.5N - 1.5N$ and with efficient attention schemes [44] N can be on the order of 8k even with moderate computational resources. For good generalization across a range of N encountered in stereo feature triangulation one would need to train GECCO on varied point cloud cardinalities, employing attention masking and padding on smaller clouds, which requires additional engineering effort. This consideration covers stereo setups, with multiview SfM resulting in orders of magnitude larger point clouds, making the problem even more challenging. Finally, the behavior of CDMs for out-of-distribution density estimation is not “hierarchical” — a locally perturbed in-distribution example are often assigned (low) probability comparable to that of examples entirely out of the domain, instead of an intuitive gradual decay of estimated probability as the scale of the perturbation grows. Concluding, future modelling work will need to solve the problems of inference speed, clouds of varying sizes and generalization as a density estimator.

4.9 Future work

While the previous section mentioned deficiencies of the current model, here we highlight research directions which may further improve the future generations of generative point cloud models.

CDM losses do not respect permutation equivariance. Some consequences of permutation equivariance for model choice have already been covered in Sec. 4.7 and addressed by prior work but the problem is still not fully solved. A topic that remains to be explored is the interaction with the diffusion loss function. As described in previous sections, in DMs the neural network is trained to denoise input data, that is given $x_{\text{noisy}} = x + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$ the loss term is

$$\mathcal{L}(x) = ||f_\theta(x_{\text{noisy}}, \sigma, \theta) - x||^2 \tag{4.8}$$

multiplied by weighing factors[2]. This formulation makes intuitive sense when x is a vector, for example an image. Each pixel is drawn to best match its original color. It however becomes problematic when x is a set. Eq. (4.9) is *not* permutation equivariant and forces the network to

[2]It is also possible to predict the noise vector ϵ rather than x and a spectrum of intermediate solutions. See [98] for discussion.

Figure 4.8: Average image in CIFAR-10, **left**: airplane, **right**: car. When σ is high a diffusion network returns the dataset average to minimize the loss in Eq. (4.9). With images, we see that airplanes are centered on blue sky, while cars are in the lower half of image frame, lit from the top and surrounded by dark ground and foliage. In contrast a point cloud model like GECCO sees no such differences between cars and airplanes at high σ.

“return” each point to its original position, even though the perturbation may have swapped out individual points. Simply put, given a perturbed pointcloud which clearly depicts an airplane it is not sufficient for the network to output *an* airplane — it must output the original shape up to the “identity” of individual points.

This informal argument is best seen in the limit of very high σ. In this regime ϵ dominates x and x_{noisy} is virtually uninformative of x, so the network is expected to minimize $\mathscr{L}$ and return the dataset average. In diffusion over vectors this average is informative of the dataset, as depicted in Fig. 4.8, and brings the sample closer to the data manifold. With point clouds however, the average needs to be taken not only over different sets of points (examples) but also over their permutations, resulting in a single value for all points. This means that diffusion models trained on (mean, variance)-normalized images of cars at $\sigma \to \infty$ will exhibit different behavior than those trained on airplanes. Meanwhile the dynamics of *all* models trained on (mean, variance)-normalized point clouds will behave identically at $\sigma \to \infty$, when clouds resemble blobs, and start to differ only at lower σ, closer to final shapes. This is important for runtime as one of the main considerations in designing CDMs is to keep the sample trajectories simple and straight to allow for bigger steps in the solvers [98, 99, 128].

Permutation equivariant losses. An obvious direction for improvement is to replace the loss in Eq. (4.9) with a permutation-invariant formulation, for example relying on chamfer distance or optimal transport losses. In fact, when a meaningful similarity measure is available for the modelled domain, optimal transport can be used to create simple GAN-like models where a network transforms a number of noise samples to data domain and is supervised to match the target (empirical) distribution in OT sense. Since the distance of individual 3D points is such a similarity measure, this suggests a possible connection beyond just respecting permutation equivariance and a potential for principled handling of the global shape variation (through the actual generative process) as well as local variation in point distribution on the object surface (through OT supervision).

In the development of GECCO we investigated replacing the loss in Eq. (4.9) with chamfer and OT equivalents. Although the issue with trivial behavior at $\sigma \rightarrow \infty$ was resolved, the resulting diffusion model did not perform as well as the standard formulation in terms of sample quality. We believe that this topic requires more in-depth consideration and can lead to substantial algorithmic improvements. In fact such work may already be underway in the broader community, with very recent methods like Flow Matching [128] pushing the field towards greater generality and more customizable diffusion schemes.

Multiview inference. Another driver of interest in diffusion models is the large number of corollary applications they bring. These include image inpainting [211, 134], latent interpolation [211], superresolution [134], colorisation [211], style transfer [94], and compositional conditional generation [129], all without needing to retrain the network specifically for the task. We show an example of such result with GECCO by adapting the inpainting method of [134] to increase the resolution of point clouds, but more can be done. In cases when the input point cloud is cut with a plane, rather than uniformly decimated, we found that the current procedure of GECCO results in a “completion” which covers the entire object rather than just the missing part. Similarly, the approach of compositional conditional generation in [129] should translate to multiview-conditioned point cloud generation. In an experiment we evaluated the score of a noisy point cloud x conditionally on multiple input views $v_{1\dots N}$ as

$$s_i = s(c_i(x), v_i, \sigma, \theta), \tag{4.9}$$

where c_i is a transformation from the reference frame of x to the coordinate system of the i-th camera. We then proceeded with the sampling scheme using an average score

$$s_{\mathrm{mv}}(x, v_{1\dots N}, \sigma, \theta) = \frac{1}{N} \sum_{i=1}^{N} c_i^{-1}(s_i). \tag{4.10}$$

Note that while x is a point cloud, s_i is a vector field and as such c_i^{-1} applies only rotation and scaling but not translation. As a whole, this means following the *mean dynamics* of each view, accounting for their relative geometry. This did not yield substantial improvements over single view conditioning on the ShapeNet-Vol [36] dataset, in which the overall view-conditional uncertainty is rather low. Additionally, this approach is not directly applicable to the Taskonomy model, since it is trained to on all points within camera’s viewing frustum (as opposed to a single centered object) and applying it on the sum or intersection of multiple viewing frustums would lead to out-of-distribution behavior of the network. While the initial attempts at these extensions were inconclusive, we believe these goals are feasible and worth exploring in future work to benefit tasks like LiDAR scan completion or uncertainty-aware multiview registration.

5 Conclusions

Throughout this thesis, the application of projective geometry to grapple with inverse problems, either directly or indirectly, has been the unifying thread. These problems are important practically because camera images lose 3D detail, which is crucial to building embodied systems or assisting humans in their everyday environment. Projective geometry, the theoretical foundation of camera image formation, is therefore a principled guide in inverting this lossy process.

First in DISK we propose new and better local features — a component of structure from motion (SfM) systems which are used to recover scene and viewing geometry from multiple images. We use the geometric definition of local features to discover them *ab initio* through an optimization process. Then in RayTran we solve such a problem ourselves, albeit in an implicit form, to build a unified latent representation of the scene and precisely locate objects of interest along with their appearances. Unlike SfM, which requires lengthy optimization for each scene, RayTran works very fast, making it applicable for large scale indoor layout detection. Finally, GECCO attacks the inverse problem most directly, predicting the geometry of a scene solely from a single image. We embed projections in the model itself, helping it maintain correspondence between the reconstructed 3D scene and the 2D input view. Because the monocular 2D to 3D setting is ill posed, we take the generative approach and instead of finding the most likely solution, or even worse the average one, we instead sample different valid geometries, reflecting the uncertainty. At the same time, GECCO provides the ability to estimate the likelihood of a given scene, making it a building block for future systems which need regularization to deal with such uncertainty.

This brings us to the other theme of this thesis — *boostrapping*. Thanks to the physics and geometry of image acquisition many problems, such as triangulation or projections, can be solved mathematically and exactly. This means that in many cases an algorithm's accuracy improves with the volume of input data. One example is SfM where even low quality traditional image matching algorithms can be used to build precise models of scenes when thousands of images are available. These are robust enough to train the next generation of data-driven algorithms, like our DISK, which match this accuracy with much fewer inputs. This process can be repeated again and again, providing diminishing but still valuable returns. A bigger bootstrapping cycle involves learning *what shapes exist in nature*, that is, constructing learnable priors over shapes like GECCO. This will enable building more precise 3D models from fewer inputs, providing more and better data for local features, object detection and next generation shape priors. Finally in Sec. 3.8.2, as a successor to RayTran, we envision a system capable of learning not just about shapes but overall geometry and appearances while avoiding distraction from uncontrollable dynamics. We believe that building such self-improvement loops is the future of 3D perception and are proud to have contributed to that end with this thesis.

Bibliography

[1] R. Achanta et al. *Slic superpixels*. Tech. rep. 2010.

[2] P. Achlioptas et al. “Learning representations and generative models for 3d point clouds”. In: *International conference on machine learning*. PMLR. 2018, pp. 40–49.

[3] S. Agarwal et al. “Building Rome in One Day”. In: *International Conference on Computer Vision*. 2009.

[4] M. Andriluka, S. Roth, and B. Schiele. “People-tracking-by-detection and people-detection-by-tracking”. In: *Conference on Computer Vision and Pattern Recognition*. 2008.

[5] R. Arandjelović and A. Zisserman. “Three things everyone should know to improve object retrieval”. In: *Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2911–2918.

[6] A. Arnab et al. “ViViT: A video vision transformer”. In: *Conference on Computer Vision and Pattern Recognition*. 2021.

[7] A. Avetisyan, A. Dai, and M. Nießner. “End-to-end cad model retrieval and 9dof alignment in 3d scans”. In: *International Conference on Computer Vision*. 2019.

[8] A. Avetisyan et al. “Scan2CAD: Learning cad model alignment in RGB-D scans”. In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[9] A. Avetisyan et al. “SceneCAD: Predicting Object Alignments and Layouts in RGB-D Scans”. In: *European Conference on Computer Vision*. 2020.

[10] V. Balntas et al. “Hpatches: A Benchmark and Evaluation of Handcrafted and Learned Local Descriptors”. In: *Conference on Computer Vision and Pattern Recognition*. 2017.

[11] A. Barroso-Laguna et al. “Key. net: Keypoint detection by handcrafted and learned cnn filters”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 5836–5844.

[12] H. Bay et al. “SURF: Speeded Up Robust Features”. In: 10.3 (2008), pp. 346–359.

[13] F. Bellavia and C. Colombo. “Is there anything new to say about SIFT matching?” In: *International Journal of Computer Vision* (2020), pp. 1–20.

[14] P. Bergmann, T. Meinhardt, and L. Leal-Taixe. “Tracking without bells and whistles”. In: *International Conference on Computer Vision*. 2019.

[15] G. Bertasius, H. Wang, and L. Torresani. “Is Space-Time Attention All You Need for Video Understanding?” In: *International Conference on Machine Learning*. 2021.

[16] P. J. Besl and N. D. McKay. “Method for registration of 3-D shapes”. In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606.

[17] A. Bhowmik et al. “Reinforced Feature Points: Optimizing Feature Detection and Description for a High-Level Task”. In: *Conference on Computer Vision and Pattern Recognition*. 2020.

[18] F. Bogo et al. “Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*. Springer. 2016, pp. 561–578.

[19] J. Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018.

[20] M. D. Breitenstein et al. “Robust tracking-by-detection using a detector confidence particle filter”. In: *International Conference on Computer Vision*. 2009.

[21] M. Brown and D. G. Lowe. “Automatic panoramic image stitching using invariant features”. In: *International Journal of Computer Vision* 74 (2007), pp. 59–73.

[22] T. Brown et al. “Language models are few-shot learners”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[23] R. Cai et al. “Doppelgangers: Learning to Disambiguate Images of Similar Structures”. In: *International Conference on Computer Vision*. 2023, pp. 34–44.

[24] R. Cai et al. “Learning gradient fields for shape generation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 364–381.

[25] N. Carion et al. “End-to-end object detection with transformers”. In: *European Conference on Computer Vision*. 2020.

[26] A. X. Chang et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv Preprint* (2015).

[27] A. Chen et al. “Tensorf: Tensorial radiance fields”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 333–350.

[28] C. Chen et al. “Unsupervised learning of fine structure generation for 3d point clouds by 2d projection matching”. In: *International Conference on Computer Vision*. 2021, pp. 12466–12477.

[29] N. Chen et al. “WaveGrad 2: Iterative refinement for text-to-speech synthesis”. In: *Interspeech* (2021).

[30] N. Chen et al. “WaveGrad: Estimating gradients for waveform generation”. In: *International Conference on Learning Representations* (2021).

[31] R. T. Chen and Y. Lipman. “Riemannian flow matching on general geometries”. In: *arXiv Preprint* (2023).

[32] R. T. Chen et al. “Neural ordinary differential equations”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[33] Z. Chen, A. Tagliasacchi, and H. Zhang. “Bsp-net: Generating compact meshes via binary space partitioning”. In: *Conference on Computer Vision and Pattern Recognition*. 2020.

[34] Z. Chen and H. Zhang. “Learning implicit fields for generative shape modeling”. In: *Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5939–5948.

[35] B. Cheng, A. G. Schwing, and A. Kirillov. “Per-Pixel Classification is Not All You Need for Semantic Segmentation”. In: *NeurIPS*. 2021.

[36] C. B. Choy et al. “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction”. In: *European Conference on Computer Vision*. 2016.

[37] O. Chum, T. Werner, and J. Matas. “Two-View Geometry Estimation Unaffected by a Dominant Plane”. In: *Conference on Computer Vision and Pattern Recognition*. 2005.

[38] T. Cieslewski, M. Bloesch, and D. Scaramuzza. “Matching Features without Descriptors: Implicitly Matched Interest Points”. In: *arXiv Preprint* (2018).

[39] T. Cieslewski, K. G. Derpanis, and D. Scaramuzza. “SIPs: Succinct interest points from unsupervised inlierness probability learning”. In: *International Conference on 3D Vision*. IEEE. 2019, pp. 604–613.

[40] M. Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3213–3223.

[41] F.-A. Croitoru et al. “Diffusion models in vision: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[42] A. Dai et al. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. In: *Conference on Computer Vision and Pattern Recognition*. 2017.

[43] A. K. Danila Rukhovich Anna Vorontsova. “ImVoxelNet: Image to Voxels Projection for Monocular and Multi-View General-Purpose 3D Object Detection”. In: *WACV*. 2022.

[44] T. Dao et al. “Flashattention: Fast and memory-efficient exact attention with io-awareness”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16344–16359.

[45] J. Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[46] D. DeTone, T. Malisiewicz, and A. Rabinovich. “Superpoint: Self-supervised interest point detection and description”. In: *Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 224–236.

[47] J. Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv Preprint* (2018).

[48] P. Dhariwal and A. Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8780–8794.

[49] L. Dinh, D. Krueger, and Y. Bengio. “Nice: Non-linear independent components estimation”. In: *International Conference on Learning Representations*. 2015.

[50] L. Dinh, J. Sohl-Dickstein, and S. Bengio. “Density estimation using real nvp”. In: *arXiv Preprint* (2016).

[51] A. Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *ICLR*. 2020.

[52] M. Dusmanu et al. “D2-Net: A Trainable CNN for Joint Detection and Description of Local Features”. In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[53] A. Duzceker et al. “DeepVideoMVS: Multi-view stereo on video with recurrent spatio-temporal fusion”. In: *Conference on Computer Vision and Pattern Recognition*. 2021.

[54] P. Ebel et al. “Beyond Cartesian Representations for Local Descriptors”. In: *International Conference on Computer Vision*. 2019.

[55] J. Edstedt et al. “DeDoDe: Detect, Don’t Describe–Describe, Don’t Detect for Local Feature Matching”. In: *arXiv preprint arXiv:2308.08479* (2023).

[56] J. Engel, V. Koltun, and D. Cremers. “Direct sparse odometry”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (2017), pp. 611–625.

[57] F. Engelmann et al. “From Points to Multi-Object 3D Reconstruction”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4588–4597.

[58] S. Ettinger et al. “Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset”. In: *International Conference on Computer Vision*. Oct. 2021, pp. 9710–9719.

[59] H. Fan, H. Su, and L. J. Guibas. “A point set generation network for 3d object reconstruction from a single image”. In: *Conference on Computer Vision and Pattern Recognition*. 2017, pp. 605–613.

[60] X. Fei and S. Soatto. “Visual-inertial object detection and mapping”. In: *European Conference on Computer Vision*. 2018.

[61] L. Freda. *PySLAM v2*. https://github.com/luigifreda/pyslam. 2020.

[62] A. Frome et al. “Recognizing objects in range data using regional point descriptors”. In: *European Conference on Computer Vision*. 2004.

[63] S. Garrido-Jurado et al. “Automatic generation and detection of highly reliable fiducial markers under occlusion”. In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292.

[64] A. Geiger, P. Lenz, and R. Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 3354–3361.

[65] K. Genova et al. “Learning 3D semantic segmentation with only 2D image supervision”. In: *International Conference on 3D Vision*. IEEE. 2021, pp. 361–372.

[66] R. Girdhar et al. “Learning a Predictable and Generative Vector Representation for Objects”. In: *European Conference on Computer Vision*. 2016.

[67] G. Gkioxari, J. Malik, and J. Johnson. “Mesh R-CNN”. In: *International Conference on Computer Vision*. 2019.

[68] P. Gleize, W. Wang, and M. Feiszli. “SiLK–Simple Learned Keypoints”. In: *arXiv Preprint* (2023).

[69] W. Grathwohl et al. “FFJORD: Free-form continuous dynamics for scalable reversible generative models”. In: *arXiv Preprint* (2018).

[70] T. Groueix et al. “A papier-mâché approach to learning 3d surface generation”. In: *Conference on Computer Vision and Pattern Recognition*. 2018, pp. 216–224.

[71] V. Guizilini et al. “Depth field networks for generalizable multi-view scene representation”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 245–262.

[72] C. Gümeli, A. Dai, and M. Nießner. “ROCA: Robust CAD Model Retrieval and Alignment from a Single Image”. In: *arXiv Preprint* (2021).

[73] X. Han et al. “MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching”. In: *Conference on Computer Vision and Pattern Recognition*. 2015.

[74] C. Harris, M. Stephens, et al. “A combined corner and edge detector”. In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[75] K. He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[76] K. He et al. “Deep residual learning for image recognition”. In: *Conference on Computer Vision and Pattern Recognition*. 2016.

[77] K. He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[78] K. He et al. “Masked autoencoders are scalable vision learners”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16000–16009.

[79] J. Heinly et al. “Reconstructing the World in Six Days”. In: *Conference on Computer Vision and Pattern Recognition*. 2015.

[80] J. Ho, A. Jain, and P. Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[81] A. G. Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv Preprint* (2017).

[82] H.-N. Hu et al. “Joint monocular 3D vehicle detection and tracking”. In: *International Conference on Computer Vision*. 2019.

[83] Q. Hu et al. “RandLA-Net: Efficient semantic segmentation of large-scale point clouds”. In: *Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11108–11117.

[84] S. Huang et al. “Holistic 3D scene parsing and reconstruction from a single RGB image”. In: *European Conference on Computer Vision*. 2018.

[85] L. Hui et al. “Progressive point cloud deconvolution generation network”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 397–413.

[86] M. F. Hutchinson. “A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines”. In: *Communications in Statistics-Simulation and Computation* 18.3 (1989), pp. 1059–1076.

[87] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International Conference on Machine Learning*. pmlr. 2015, pp. 448–456.

[88] C. Ionescu et al. “Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments”. In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013), pp. 1325–1339.

[89] H. Izadinia and S. M. Seitz. “Scene Recomposition by Learning-based ICP”. In: *Conference on Computer Vision and Pattern Recognition*. 2020.

[90] H. Izadinia, Q. Shan, and S. M. Seitz. “Im2CAD”. In: *Conference on Computer Vision and Pattern Recognition*. 2017.

[91] A. Jabri et al. “DORSal: Diffusion for Object-centric Representations of Scenes et al”. In: *arXiv Preprint* (2023).

[92] A. Jain et al. “Zero-shot text-guided object generation with dream fields”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 867–876.

[93] W. Jakob et al. *Mitsuba 3 renderer*. Version 3.0.1. https://mitsuba-renderer.org. 2022.

[94] J. Jeong, M. Kwon, and Y. Uh. “Training-free Style Transfer Emerges from h-space in Diffusion models”. In: *arXiv Preprint* (2023).

[95] Y. Jin et al. “Image Matching across Wide Baselines: From Paper to Practice”. In: *International Journal of Computer Vision* (2020).

[96] K. Joseph et al. “Towards open world object detection”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5830–5840.

[97] A. Karpathy. *MinGPT*. https://github.com/karpathy/minGPT/. 2020.

[98] T. Karras et al. “Elucidating the Design Space of Diffusion-Based Generative Models”. In: *arXiv Preprint* (2022).

[99] J. Kelly et al. “Learning differential equations that are easy to solve”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4370–4380.

[100] B. Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4 (2023).

[101] P. Kidger. “On Neural Differential Equations”. PhD thesis. University of Oxford, 2021.

[102] P. Kidger and C. Garcia. “Equinox: neural networks in JAX via callable PyTrees and filtered transformations”. In: *Differentiable Programming workshop at Neural Information Processing Systems 2021* (2021).

[103] H. Kim et al. “Softflow: Probabilistic framework for normalizing flow on manifolds”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16388–16397.

[104] J. Kim et al. “SetVAE: Learning hierarchical composition for generative modeling of set-structured data”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15059–15068.

[105] D. Kingma et al. “Variational diffusion models”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 21696–21707.

[106] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *International Conference on Learning Representations*. 2015.

[107] R. Klokov, E. Boyer, and J. Verbeek. “Discrete point flow networks for efficient point cloud generation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 694–710.

[108] Z. Kong et al. “Diffwave: A versatile diffusion model for audio synthesis”. In: *International Conference on Learning Representations* (2021).

[109] A. Krizhevsky, G. Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[110] A. Kundu, Y. Li, and J. M. Rehg. “3D-RCNN: Instance-level 3d object reconstruction via render-and-compare”. In: *Conference on Computer Vision and Pattern Recognition*. 2018.

[111] W. Kuo et al. “Mask2CAD: 3D Shape Prediction by Learning to Segment and Retrieve”. In: *European Conference on Computer Vision*. 2020.

[112] W. Kuo et al. “Patch2CAD: Patchwise Embedding Learning for In-the-Wild Shape Retrieval from a Single Image”. In: *International Conference on Computer Vision*. 2021.

[113] A. B. Laguna et al. “Key. net: Keypoint detection by handcrafted and learned cnn filters”. In: *International Conference on Computer Vision*. 2019.

[114] K. Lasinger et al. “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer”. In: *arXiv Preprint* (2019).

[115] Y. LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[116] J. Lee et al. “Set transformer: A framework for attention-based permutation-invariant neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 3744–3753.

[117] R. Leroy et al. “Pix2point: Learning outdoor 3d using sparse point clouds and optimal transport”. In: *2021 17th International Conference on Machine Vision and Applications (MVA)*. IEEE. 2021, pp. 1–5.

[118] T. Lewiner et al. “Efficient Implementation of Marching Cubes’ Cases with Topological Guarantees”. In: *J. Graphics, GPU, & Game Tools* 8.2 (2003), pp. 1–15.

[119] C.-L. Li et al. “Point cloud gan”. In: *arXiv Preprint* (2018).

[120] K. Li, H. Rezatofighi, and I. Reid. “MOLTR: Multiple Object Localization, Tracking and Reconstruction From Monocular RGB Videos”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3341–3348.

[121] K. Li et al. “ODAM: Object Detection, Association, and Mapping using Posed RGB Video”. In: *International Conference on Computer Vision*. 2021.

[122] R. Li et al. “SP-GAN: Sphere-guided 3D shape generation and manipulation”. In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–12.

[123] Y. Li et al. “Database-Assisted Object Retrieval for Real-Time 3D Reconstruction”. In: *Computer Graphics Forum*. Vol. 34. Wiley Online Library. 2015.

[124] Z. Li and N. Snavely. “MegaDepth: Learning Single-View Depth Prediction from Internet Photos”. In: *Conference on Computer Vision and Pattern Recognition*. 2018.

[125] Y. Liao, J. Xie, and A. Geiger. “KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[126] T.-Y. Lin et al. “Microsoft coco: Common objects in context”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[127] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys. “LightGlue: Local Feature Matching at Light Speed”. In: *ICCV*. 2023.

[128] Y. Lipman et al. “Flow matching for generative modeling”. In: *arXiv Preprint* (2022).

[129] N. Liu et al. “Compositional visual generation with composable diffusion models”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 423–439.

[130] Z. Liu et al. “Point-voxel CNN for efficient 3d deep learning”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[131] Z. Liu et al. “A convnet for the 2020s”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11976–11986.

[132] I. Loshchilov and F. Hutter. “Decoupled weight decay regularization”. In: *arXiv Preprint* (2017).

[133] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 20.2 (Nov. 2004), pp. 91–110.

[134] A. Lugmayr et al. “Repaint: Inpainting using denoising diffusion probabilistic models”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11461–11471.

[135] S. Luo and W. Hu. “Diffusion probabilistic models for 3d point cloud generation”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2837–2845.

[136] Z. Luo et al. “Contextdesc: Local Descriptor Augmentation with Cross-Modality Context”. In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[137] Z. Luo et al. “Geodesc: Learning Local Descriptors by Integrating Geometry Constraints”. In: *European Conference on Computer Vision*. 2018.

[138] S. Lynen et al. “Large-scale, real-time visual-inertial localization revisited”. In: *International Journal of Robotics Research* (2020).

[139] Z. Lyu et al. “A conditional point diffusion-refinement paradigm for 3d point cloud completion”. In: *arXiv Preprint* (2021).

[140] S. Mahendran, H. Ali, and R. Vidal. “A mixed classification-regression framework for 3d pose estimation from 2d images”. In: *arXiv Preprint* (2018).

[141] K.-K. Maninis et al. “Vid2CAD: Cad model alignment using multi-view constraints from videos”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[142] T. Meinhardt et al. “Trackformer: Multi-object tracking with transformers”. In: *arXiv Preprint* (2021).

[143] L. Mescheder et al. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[144] L. Mescheder et al. “Occupancy networks: Learning 3d reconstruction in function space”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4460–4470.

[145] L. Mescheder et al. “Occupancy networks: Learning 3d reconstruction in function space”. In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[146] K. Mikolajczyk and C. Schmid. “Scale and Affine Invariant Interest Point Detectors”. In: *International Journal of Computer Vision* 60 (2004), pp. 63–86.

[147] B. Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[148] Y. Ming et al. “Deep learning for monocular depth estimation: A review”. In: *Neurocomputing* 438 (2021), pp. 14–33.

[149] A. Mishchuk et al. “Working Hard to Know Your Neighbor’s Margins: Local Descriptor Learning Loss”. In: *Advances in Neural Information Processing Systems*. 2017.

[150] D. Mishkin, F. Radenovic, and J. Matas. “Repeatability is Not Enough: Learning Affine Regions via Discriminability”. In: *European Conference on Computer Vision*. 2018.

[151] H. P. Moravec. “Techniques towards automatic visual obstacle avoidance”. In: (1977).

[152] A. Mousavian et al. “3d bounding box estimation using deep learning and geometry”. In: *Conference on Computer Vision and Pattern Recognition*. 2017.

[153] T. Müller et al. “Instant neural graphics primitives with a multiresolution hash encoding”. In: *ACM Transactions on Graphics (ToG)* 41.4 (2022), pp. 1–15.

[154] R. Mur-Artal, J. Montiel, and J. Tardós. “Orb-Slam: A Versatile and Accurate Monocular Slam System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[155] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* (2015).

[156] L. Nan, K. Xie, and A. Sharf. "A search-classify approach for cluttered indoor scene understanding". In: *ACM Transactions on Graphics (TOG)* (2012).

[157] NASA. *Mars Global Surveyor images of the Cydonia Region of Mars*. Accessed October 25, 2023. https://nssdc.gsfc.nasa.gov/planetary/mgs_cydonia.html. 1976, 1996.

[158] L. Nicholson, M. Milford, and N. Sünderhauf. "Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam". In: *RA-L* 4.1 (2018), pp. 1–8.

[159] Y. Nie et al. "Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes from a Single Image". In: *Conference on Computer Vision and Pattern Recognition*. 2020.

[160] H. Noh et al. "Large-scale image retrieval with attentive deep local features". In: *International Conference on Computer Vision*. 2017, pp. 3456–3465.

[161] Y. Ono et al. "Lf-Net: Learning Local Features from Images". In: *Advances in Neural Information Processing Systems*. 2018.

[162] J. J. Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174.

[163] J. J. Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[164] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[165] M. Pollefeys, R. Koch, and L. Van Gool. "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters". In: *IJCV* 32.1 (1999), pp. 7–25.

[166] B. Poole et al. "Dreamfusion: Text-to-3d using 2d diffusion". In: *arXiv Preprint* (2022).

[167] S. Popov, P. Bauszat, and V. Ferrari. "CoReNet: Coherent 3D scene reconstruction from a single RGB image". In: *European Conference on Computer Vision*. 2020.

[168] V. Popov et al. "Grad-tts: A diffusion probabilistic model for text-to-speech". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8599–8608.

[169] J. Postels et al. "Go with the flows: Mixtures of normalizing flows for point cloud generation and reconstruction". In: *International Conference on 3D Vision*. IEEE. 2021, pp. 1249–1258.

[170] A. Pumarola et al. "C-flow: Conditional generative flow models for images and 3d point clouds". In: *Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7949–7958.

[171] C. R. Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[172] C. R. Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[173] G. Qian et al. “PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies”. In: *Advances in Neural Information Processing Systems* (2022).

[174] S. Qian, L. Jin, and D. F. Fouhey. “Associative3D: Volumetric Reconstruction from Sparse Views”. In: *European Conference on Computer Vision*. 2020.

[175] A. Radford et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[176] S. Ramasinghe and S. Lucey. “Beyond periodicity: Towards a unifying framework for activations in coordinate-MLPs”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 142–158.

[177] A. Ramesh et al. “Hierarchical text-conditional image generation with clip latents”. In: *arXiv Preprint* (2022).

[178] N. Ravi et al. “Accelerating 3D Deep Learning with PyTorch3D”. In: *arXiv Preprint* (2020).

[179] A. Razavi, A. Van den Oord, and O. Vinyals. “Generating diverse high-fidelity images with vq-vae-2”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[180] J. Redmon et al. “You only look once: Unified, real-time objec/bio detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[181] J. Reizenstein et al. “Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10901–10911.

[182] S. Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *NIPS*. 2015.

[183] J. Revaud et al. “R2D2: Repeatable and Reliable Detector and Descriptor”. In: *arXiv Preprint*. 2019.

[184] D. Rezende and S. Mohamed. “Variational inference with normalizing flows”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1530–1538.

[185] E. Riba et al. “Kornia: an Open Source Differentiable Computer Vision Library for PyTorch”. In: *WACV*. 2020.

[186] I. Rocco et al. “Neighbourhood Consensus Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[187] R. Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[188] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Conference on Medical Image Computing and Computer Assisted Intervention*. 2015, pp. 234–241.

[189] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[190] M. Runz et al. “FroDO: From Detections to 3D Objects”. In: *Conference on Computer Vision and Pattern Recognition*. 2020.

[191] A. Safin, D. Durckworth, and M. S. Sajjadi. “RePAST: Relative Pose Attention Scene Representation Transformer”. In: *arXiv Preprint* (2023).

[192] C. Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *Advances in Neural Information Processing Systems* (2022).

[193] M. S. Sajjadi et al. “Object scene representation transformer”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9512–9524.

[194] M. S. Sajjadi et al. “Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6229–6238.

[195] R. F. Salas-Moreno et al. “SLAM++: Simultaneous localisation and mapping at the level of objects”. In: *Conference on Computer Vision and Pattern Recognition*. 2013.

[196] A. Sanghi et al. “Clip-forge: Towards zero-shot text-to-shape generation”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18603–18613.

[197] E. Santellani et al. “S-TREK: Sequential Translation and Rotation Equivariant Keypoints for local feature extraction”. In: *Conference on Computer Vision and Pattern Recognition*. 2023, pp. 9728–9737.

[198] P.-E. Sarlin et al. “SuperGlue: Learning Feature Matching with Graph Neural Networks”. In: *Conference on Computer Vision and Pattern Recognition* (2020).

[199] T. Sattler et al. “Understanding the limitations of CNN-based absolute camera pose regression”. In: *Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3302–3312.

[200] N. Savinov et al. “Quad-Networks: Unsupervised Learning to Rank for Interest Point Detection”. In: *Conference on Computer Vision and Pattern Recognition*. 2017.

[201] A. Sax et al. “Learning to Navigate Using Mid-Level Visual Priors”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 791–812.

[202] J. L. Schönberger and J.-M. Frahm. “Structure-from-motion revisited”. In: *Conference on Computer Vision and Pattern Recognition*. 2016.

[203] J. L. Schönberger et al. “Comparative Evaluation of Hand-Crafted and Learned Local Features”. In: *Conference on Computer Vision and Pattern Recognition*. 2017.

[204] J. L. Schönberger et al. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision*. 2016.

[205] M. Shan et al. “ELLIPSDF: Joint Object Pose and Shape Optimization with a Bi-level Ellipsoid and Signed Distance Function Description”. In: *International Conference on Computer Vision*. 2021.

[206] T. Shao et al. “An interactive approach to semantic modeling of indoor scenes with an RGBD camera”. In: *ACM Transactions on Graphics (TOG)* (2012).

[207] D. W. Shu, S. W. Park, and J. Kwon. “3d point cloud generative adversarial network based on tree structured graph convolutions”. In: *Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3859–3868.

[208] E. Simo-Serra et al. “Discriminative Learning of Deep Convolutional Feature Point Descriptors”. In: *International Conference on Computer Vision*. 2015.

[209] M. Skorski. “Modern Analysis of Hutchinson’s Trace Estimator”. In: *2021 55th Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2021, pp. 1–5.

[210] J. Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.

[211] Y. Song et al. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *International Conference on Learning Representations*. 2021.

[212] R. Strudel et al. “Segmenter: Transformer for Semantic Segmentation”. In: *International Conference on Computer Vision*. 2021.

[213] J. Sun et al. “LoFTR: Detector-free local feature matching with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 8922–8931.

[214] W. Sun et al. “Acne: Attentive context normalization for robust permutation-equivariant learning”. In: *Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11286–11295.

[215] Y. Sun et al. “Pointgrow: Autoregressively learned point cloud generation with self-attention”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 61–70.

[216] J. Tae. “Mirror Diffusion Models”. In: *arXiv Preprint* (2023).

[217] Y. Tian, B. Fan, and F. Wu. “L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space”. In: *Conference on Computer Vision and Pattern Recognition*. 2017.

[218] Y. Tian et al. “SOSNet: Second order similarity regularization for local descriptor learning”. In: *Conference on Computer Vision and Pattern Recognition*. 2019.

[219] P. Truong et al. “GLAMpoints: Greedily Learned Accurate Match points”. In: *International Conference on Computer Vision*. 2019.

[220] S. Tulsiani et al. “Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene”. In: *Conference on Computer Vision and Pattern Recognition*. 2018.

[221] M. J. Tyszkiewicz, P. Fua, and E. Trulls. “DISK: Learning local features with policy gradient”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14254–14265.

[222] M. J. Tyszkiewicz, P. Fua, and E. Trulls. “Gecco: Geometrically-conditioned point diffusion models”. In: *International Conference on Computer Vision* (2023).

[223] M. J. Tyszkiewicz et al. “RayTran: 3D pose estimation and shape reconstruction of multiple objects from videos with ray-traced transformers”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 211–228.

[224] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv Preprint* (2016).

[225] A. Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[226] Y. Verdie et al. “TILDE: A Temporally Invariant Learned DEtector”. In: *Conference on Computer Vision and Pattern Recognition*. 2015.

[227] N. Wang et al. “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”. In: *European Conference on Computer Vision*. 2018.

[228] Q. Wang et al. “Learning feature descriptors using camera pose supervision”. In: *European Conference on Computer Vision* (2020).

[229] R. J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* (1992).

[230] C. Wu. “Towards linear-time incremental structure from motion”. In: *3DV*. 2013.

[231] J. Wu et al. “Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling”. In: *NIPS*. 2016.

[232] J. Wu et al. “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling”. In: *Advances in Neural Information Processing Systems* 29 (2016).

[233] Y. Wu and K. He. “Group normalization”. In: *International Conference on 3D Vision*. 2018, pp. 3–19.

[234] H. Xie et al. “Pix2Vox++: multi-scale context-aware 3D object reconstruction from single and multiple images”. In: *IJCV* 128.12 (2020), pp. 2919–2935.

[235] J. Xie et al. “Generative PointNet: Deep energy-based learning on unordered point sets for 3d generation, reconstruction and classification”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14976–14985.

[236] G. Yang et al. “Pointflow: 3d point cloud generation with continuous normalizing flows”. In: *Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4541–4550.

[237] S. Yang and S. Scherer. “Cubeslam: Monocular 3-d object slam”. In: *IEEE Transactions on Robotics* 35.4 (2019), pp. 925–938.

[238] K. M. Yi et al. “Learning to Assign Orientations to Feature Points”. In: *Conference on Computer Vision and Pattern Recognition*. 2016.

[239] K. M. Yi et al. “Learning to Find Good Correspondences”. In: *Conference on Computer Vision and Pattern Recognition*. 2018.

[240] K. M. Yi et al. “LIFT: Learned Invariant Feature Transform”. In: *European Conference on Computer Vision*. 2016.

[241] K. M. Yi et al. “Lift: Learned invariant feature transform”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 467–483.

[242] S. Zagoruyko and N. Komodakis. “Learning to Compare Image Patches via Convolutional Neural Networks”. In: *Conference on Computer Vision and Pattern Recognition*. 2015.

[243] A. R. Zamir et al. “Taskonomy: Disentangling task transfer learning”. In: *Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3712–3722.

[244] X. Zeng et al. “LION: Latent Point Diffusion Models for 3D Shape Generation”. In: *Advances in Neural Information Processing Systems* (2022).

[245] D. Zhang et al. “Learning to generate 3d shapes with generative cellular automata”. In: *arXiv Preprint* (2021).

[246] J. Zhang et al. “Learning two-view correspondences and geometry using order-aware network”. In: *Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5845–5854.

[247] H. Zhao et al. “Point transformer”. In: *International Conference on Computer Vision*. 2021, pp. 16259–16268.

[248] X. Zhao et al. “ALIKED: A Lighter Keypoint and Descriptor Extraction Network via Deformable Transformation”. In: *IEEE Transactions on Instrumentation and Measurement* (2023).

[249] S. Zheng et al. “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers”. In: *Conference on Computer Vision and Pattern Recognition*. 2021.

[250] L. Zhou, Y. Du, and J. Wu. “3d shape generation and completion through point-voxel diffusion”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5826–5835.

[251] X. Zhu et al. “Cylindrical and asymmetrical 3d convolution networks for lidar segmentation”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9939–9948.

[252] J. Zhuang et al. “Adabelief optimizer: Adapting stepsizes by the belief in observed gradients”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18795–18806.

Michał Jan Tyszkiewicz

Born: 14 June 1994 **E-mail**: michal.tyszkiewicz@gmail.com **Website**: jatentaki.github.io

Publications and conferences [Google scholar]

10.2023	GECCO: Geometrically-Conditioned Point Diffusion Models, *ICCV 2023*
09.2023	The use of machine learning in the identification of archaeological sites in the area of the Polish lowland, *UISPP XX World Congress*
10.2022	RayTran: 3D pose estimation and shape reconstruction of multiple objects from videos with ray-traced transformers, *ECCV 2022*
12.2020	DISK: learning local features with policy gradient, *NeurIPS 2020*, spotlight
08.2015	Salts of highly fluorinated weakly coordinating anions as versatile precursors towards hydrogen storage materials, *Dalton Transactions*

Research experience

PhD program in computer science at École Polytechnique Fédérale de Lausanne,
advised by prof. Pascal Fua, coadvised by Eduard Trulls (Google Zurich)

09.2019 – 12.2023	Image-conditional point cloud generation with diffusion models 3d object detection and meshing from video, using sparse transformers Transformer-based image matching Learning local feature detection and description via policy gradient (RL)

Master's degree at École Polytechnique Fédérale de Lausanne, advised by prof. Pascal Fua

Fall 2018	**Master project**: Exploiting rotation equivariance in 2 and 3d image segmentation Extension of harmonic networks, work with 3d steerable CNNs Segmentation of biomedical datasets with derivatives of the unet architecture
Fall 2017	**Semester project**: Separation of time dependent Raman spectra by means of non-negative matrix factorization
Fall 2016	**Semester project**: Blind deconvolution of multi-spectral astronomical images

BSc researcher at the LTNFM, University of Warsaw. Group leader prof. Wojciech Grochala

2015-2016	**Contracted work**: research on reactivity of $AgSO_4$ towards organic compounds
2013-2015	**Bachelor project**: synthesis of $K_2[Mn(BH_4)_4]$ with a novel patented protocol Unstable compounds: work in argon-filled glovebox, liquid nitrogen cooling

Programming skills

Languages	Python, C++, Java, Rust, basic JS and TS [StackOverflow]
Experience with major libraries/frameworks	**ML/Scientific**: PyTorch, JAX, Scipy, Numpy **Parallel/GPU**: CUDA, WebGPU [NeRF renderer]
Contests	**Hashcode**: 326/10745 in **2020**, 395/6640 in **2019**

Education, university courses

2019-2023	**PhD program in Computer Science, École Polytechnique Fédérale de Lausanne** • Multi-agent systems, ray tracing graphics, MCMC methods • Machine learning for natural sciences (focus on chemistry)
2016-2019	**MSc in Computational Science and Engineering, École Polytechnique Fédérale de Lausanne** • Numerical analysis, computational linear algebra • Signal and image processing, information theory, spiking neural networks • Approximation algorithms, linear programming • Software engineering
2013-2016	**BSc in Chemistry, supplemented with applied mathematics and computer science. College of Inter-Faculty Individual Studies in Mathematics and Natural Sciences, University of Warsaw** • Organic/inorganic/physical/quantum chemistry, physics, crystallography • Linear algebra, calculus, differential equations, probability theory • Programming, algorithmics, automata theory

Work experience

09.2019 – 06.2023	Teaching duties in CV and Intro to Programming, part of the PhD program at EPFL
06.2021- 11.2021	PhD researcher intern at **Google Zurich**, Vitto Ferrari's team. **Topic**: 3d object detection from posed images **Tools**: Docker, PyTorch, Google Cloud Platform
02.2018 – 07.2018	Software engineering intern at **Nvidia** (Santa Clara), Sean Pieper's team. **Topic**: Development of physically-realistic postprocessing pipeline for simulating camera acquisition artifacts on synthetic imagery **Tools**: CUDA, C++, Python, OpenCV
09.2017 – 01.2018	Student teaching assistant in Software Engineering, EPFL
09.2015 – 03.2016	Researcher at Centre for New Technologies (CeNT), University of Warsaw **Topic**: Reactivity of $AgSO_4$ towards 2,6 and 2,7-ditertbutylnaphthalene

Languages

Native **Polish**, fluent **English**, intermediate **German**, basic **French** and **Russian**

Interests and activities

Travelling	Bicycle tours: Warsaw-Tallin, Warsaw-Sofia-Warsaw, Warsaw-Istanbul-Batumi
Astronomy	Project at EPFL; translations for Wikipedia [link]